이학박사 학위논문

Adaptive Image Processing Based on Superpixel

(슈퍼픽셀을 이용한 적응형 이미지 처리)

2020년 2월

서울대학교 대학원

수리과학부

우 동 준

Adaptive Image Processing Based on Superpixel

(슈퍼픽셀을 이용한 적응형 이미지 처리)

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2019년 10월

서울대학교 대학원

수리과학부

우 동 준

우 동 준의 이학박사 학위논문을 인준함

2019년 12월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Adaptive Image Processing Based on Superpixel

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Woo Dongjun

Dissertation Director : Professor Kang Myungju

Department of Mathematical Sciences
Seoul National University

February 2020

Abstract

Human-like Artificial Intelligence(AI) is also crucial issue in the field of image processing. In this trend, classical methods, processing images based on each pixel show some limitation because human being don't focus information of a single pixel. As recent studies shows, humans interpret images as the complicated combination of the number of meaningful 'clusters'. Therefore in order to deal with various and complex images in the human-like way, we should process images based on these 'clusters'.

This paper will cover superpixels that can act as these 'clusters' in images. We will introduce several superpixel generating algorithms and their advantages and disadvantages. And we will show the effectiveness of the superpixels in the image processing based on their contribution to the image evaluation field. Next, we propose a new approach to image analysis based on pivot colors of them. To find pivot colors, we propose a novel method called Superpixelwise Mean shift. This method combines the idea of mean shift procedure and the representative superpixels and is fast and robust. In the latter chapters, we will show its application to the image segmentation problem and the color mapping problem and result of them.

Key words: Image Processing, Superpixel, Clustering, Image Segmentation, Color Mapping
Student Number: 2015-30088

Contents

Abstract **i**

1 Introduction **1**

2 Preliminaries **4**

2.1 Superpixel Generating Methods 5

2.1.1 Various Superpixel Generating Methods 5

2.1.2 Performance Comparison between the Superpixels and Choosing the Method 16

2.2 Image Quality Assessment System and Superpixels 17

2.2.1 Object of Image Evaluation System 17

2.2.2 Various Image Quality Assessment System 18

2.2.3 Applying Superpixels to IQA System 24

2.2.4 Performance Comparison of IQA System 28

3 Adaptive Image Segmentation Based on Superpixel **30**

3.1 Superpixelwise Mean Shift 31

3.2 Two-Step Approach using S-Mean Shift 36

3.3 Gradient Transition and Eliminating Small Pieces 39

3.4 Merging On Balanced Gradient Transition 42

3.5 Experimental Result 45

3.5.1 Experiment of CSIQ Dataset 45

3.5.2 Experiment of Berkeley Dataset 50

3.5.3 Computational Time and Parameters 55

CONTENTS

4 Color Mapping Based on Superpixel **57**
4.1 Color Mapping Problem and Superpixels 58
4.2 Applying S-Mean Shift to Color Mapping 63

5 Conclusion **67**

Abstract (in Korean) **73**

Acknowledgement (in Korean) **74**

Chapter 1

Introduction

Human-like Artificial Intelligence(AI) is a crucial issue in the various fields. In this trend, an indistinguishable result from man's own doing is also demanded to the image processing. For human-like image processing, the study on Human Visual System(HVS) in the field of image evaluation is greatly helpful. As shown in chapter 2, from the classical Structural Similarity(SSIM) [14] to recent Superpixelwise Structural Similarity(SPSIM) [18], it is evident that human beings pay more attention to the overall structure of images than to the local information of each pixel. Therefore, we need a novel frame to obtain more human-like results based on their structure. This frame should use global properties of images not local, and should not take too long time even for large images.

In the analysis of overall structure of image, we notice that the pivot colors which dominate the images play the key role. This colors determines overall impression of the images and have a great influence to the various image processing. We call these colors as 'Representative Color'(RC) of the image and to deal with the image adaptively, procedure finding these colors should be preceded. But finding these values from the raw pixel data is not an good idea. For usual 512×512 images, we should deal with about 260,000 pixels and it will take too long time for pre-processing. Also, this approach is very vulnerable to noises such as salt and pepper. To avoid these problems, proper simplification methods are essential. By reducing the elements in the

images and cutting back on unnecessary calculations, we can handle larger and more complex images.

For this purpose, superpixel generation methods can be a good candidate. Superpixel methods, start from some over-segmentation method, combine connected pixels of similar low-level features such as coordinate and RGB values, to make 'Big Pixel's and drastically reduce the elements in the images. Each pixel is contained in one and only one superpixel and some researchers refer to only those as superpixel algorithms that can control the number and properties of superpixels such as compactness and minimum size to distinguish from over-segmentation methods. We refer these algorithms or individual 'Big Pixel's as superpixel. Classical superpixel algorithms such as the Normalized Cut(NCut) [2] algorithm and the Mean Shift Segmentation [5] algorithm have weaknesses in speed and therefore have limited usage as simplification method. But in the last decade, as fast algorithms such as Simple Iterative Clustering(SLIC) [8] method and Linear Spectral Clustering(LSC) [7] method emerge, it is now possible to apply to a wider range of fields. Not just speed, the development of superpixel methods has been spectacular over the past decade. As Stutz, Hermans and Leibe [10] shows in 2018, recent methods show much higher boundary adherence with faster speed. We will introduce the details of some major superpixel methods in the chapter 2.

The Figure 1.1 is the example of the superpixel generation. The image is the '1600.png' from the CSIQ [21] dataset. Superpixels are generated using Simple Linear Iterative Clustering(SLIC) [8] method, one of the most popular superpixel methods. We get 3,300 segments from the 512×512 image. The original image is composed of 262,144 pixels, so the object we should handle reduced to 1.157% while preserving most edges of the images. It takes about 0.5 seconds while the classical mean shift algorithm takes one minute for 256×256 sized 'cameraman.tif.' While reducing the number of objects, it preserves most edges of the images. Moreover, it eliminates the effect of the local noises and meaningless fine textures like grasses in the field and makes our methods more robust.

There have been some attempt to apply superpixels to various problems. Ko and Ding [32] proposed segmentation algorithm using superpixels in 2016,

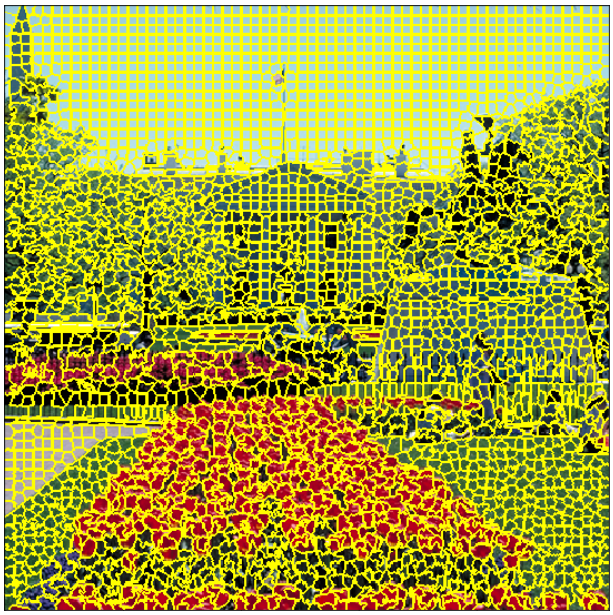


Figure 1.1: Example of Generated Superpixels

and Giraud, Ta and Papadakis [30] proposed a color transfer method based on superpixel patches in 2017. Not only in classical image processing, Dunnings and Breckon [33] try to adopt superpixel ideas to their convolutional neural network in 2018. But all these methods focus the neighboring relation between superpixels, not global properties of images. In this paper, we will introduce a new methodology called 'Superpixelwise Mean Shift'(S-Mean Shit) to grasp the color structure of images. And based on this, we perform image segmentation and color mapping. For various and massive data set, this method requires little parameter setting and no training.

In the next chapter, we will introduces the concept of the superpixel, and covers the generation of the superpixels and their coincidence to human visual system. In chapter 3, we propose S-Mean Shift and their application to image segmentation. In chapter 4, we try to apply S-mean shift to color mapping problem. Finally, we conclude our paper and propose the future works in the chapter 5.

Chapter 2

Preliminaries

Since all the proposed methods in this paper are based on superpixels, we would like to provide some background knowledge about superpixels, beforehand to help understand the latter chapters. As mentioned previous chapter, superpixel algorithms combine nearby similar pixels and generates 'Big pixel's. There are a lot of superpixel algorithms and many of them are proposed in the last decade. In this chapter, we introduce some of them and their applications.

In section 2.1, we cover famous superpixel generating algorithms first. Some are directly adopted in our algorithm, some influence our methodology and others are useful to understand the history and the effect of the superpixels. For every algorithm, we attached a result example of the algorithm made by ourselves. Finally, we compare the performance of these algorithms and propose the reason why we adopt the simple linear iterative clustering algorithm in this paper based on speed and adherence.

In section 2.2, we cover the history of the Image Quality Assessment(IQA) systems moving from individual pixel information to structural information. By tracking the development of the IQA system, we get a profound insight into the Human Visual System(HVS). And next, we introduce the contribution of the superpixels in this field. Based on the results of applying superpixels to the IQA system, one can gain assurance that the concept of the superpixel is very similar to the human cognitive system.

2.1 Superpixel Generating Methods

2.1.1 Various Superpixel Generating Methods

Before using superpixels, we should mention how to generate them. There are a lot of methods that generate superpixels, each of which has its own characteristics and advantages. It is crucial to understand these points in order to skillfully utilize superpixels. We are here to introduce some of the major superpixel generating methods before entering the main topics.

One big category for superpixel generating methods is graph-based methods. These methods turn images into graphs of pixels. Each pixel becomes a vertex and edges are connecting adjacent pixels. Weights of the edges are determined based on the similarity of the two pixels connected by those edges. The methods in this classification generally show relatively better boundary adherence, but are much slower than other categories since turning images into graphs are a time-consuming process.

Normalized cut(Ncut) [2] method is one of the earliest superpixel methods and using the global property to oversegment images. It is the classic graph-based superpixel generating method. In this method, each pixel is a vertex of the graph, and edges are connecting nearby pixels. The weight of the edges is determined as below.

$$W_{ij} = e^{-\frac{||F(i)-F(j)||_2^2}{\sigma_I^2}} \cdot \begin{cases} e^{-\frac{||X(i)-X(j)||_2^2}{\sigma_X^2}} & \text{if } ||X(i)-X(j)||_2 < r \\ 0 & \text{otherwise} \end{cases} \tag{2.1.1}$$

where $F(i)$ means the color of the ith pixel and $X(i)$ means the coordinate of the ith pixel.

We want to divide the $G = (E, V)$ into two connected subgraphs A and B such that the pixels in the same subgraph as similar as possible, and the pixels in the other subgraph as different as possible. To do this, Wu and Leahy [1] proposed a method using the cut of the two subgraphs as defined below.

$$cut(A, B) := \Sigma_{u \in A, v \in B} W(u, v) \tag{2.1.2}$$

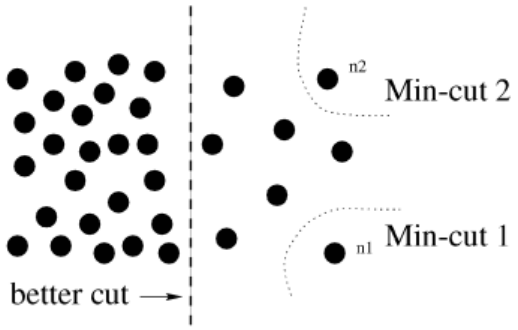


Figure 2.1: Solution of the Minimum Cut and Desired Cut [2]

But when we find A and B that make the cut minimum, we often get extremely biased results as Figure 2.1. This cut is minimized when we cutting small sets of isolated vertices in the graph. Even in the works of Wu and Leashy mentioned this phenomenon.

To solve this problem, Shi and Malik proposed the Normalized cut method. Using the Normalized cut below, we will get more balanced subgraphs.

$$Ncut(A,B) := \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(B,A)}{assoc(B,V)} \tag{2.1.3}$$

where $assoc(A,V) := \Sigma_{u\in A, v\in V} W(u,v)$. But it is proved that solving this problem is NP-complete. This is shown in the Appendix of the [2]. So we need to heuristic method to solve this problem for real-valued discrete case.

Let $x_i := \begin{cases} 1 & \text{if node } i \in A \\ -1 & \text{otherwise} \end{cases}$, and $d(i) := \Sigma_j w(i,j)$.

Using the notations above, NCut can be written as below:

$$Ncut(A,B) := \frac{-\Sigma_{x_i>0, x_j<0} w_{ij} x_i x_j}{\Sigma_{x_i>0} d(i)} + \frac{-\Sigma_{x_i<0, x_j>0} w_{ij} x_i x_j}{\Sigma_{x_i<0} d(i)} \tag{2.1.4}$$

If we define N, D, W and k as $\begin{cases} N & := |V| \\ D & := N \times N \text{ diagonal matrix with } D_{i,i} = d_i \\ W & := N \times N \text{matrix with } W_{i,j} = w_{ij} \\ k & := \frac{\Sigma_{x_i>0} d_i}{\Sigma d_i} \end{cases}$

normalized cut can be rewritten as matrix form.

$$\begin{aligned} 4 \cdot NCut(A,B) =& \frac{(1+x)^T(D-W)(1+x)}{k1^TD1} + \frac{(1-x)^T(D-W)(1-x)}{(1-k)1^TD1} \\ =& \frac{x^T(D-W)x + 1^T(D-W)1}{k(1-k)1^TD1} + \frac{2(1-2k)1^T(D-W)1}{k(1-k)1^TD1} \end{aligned} \tag{2.1.5}$$

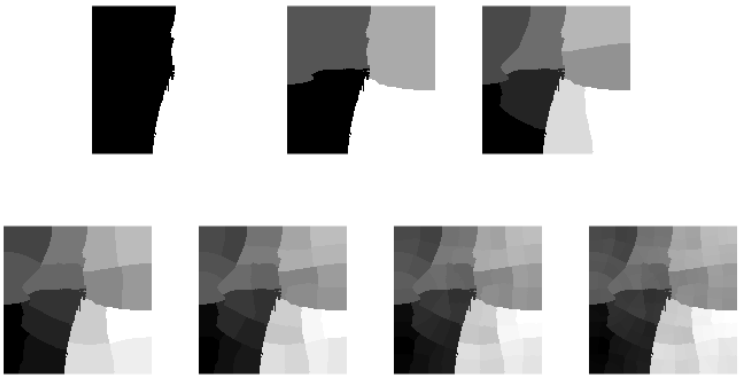
If we define $\begin{cases} \alpha(x) & := x^T(D-W)x \\ \beta(x) & := 1^T(D-W)x \\ \gamma(x) & := 1^T(D-W)1 \\ M & := 1^TD1 \end{cases}$, then the equation is rewritten as

$$\begin{aligned} =& \frac{(\alpha+\gamma) + 2(1-2k)\beta}{k(1-k)M} \\ =& \frac{(1-2k+2k^2)(\alpha+\gamma) + 2(1-2k)\beta}{k(1-k)M} + \frac{2\alpha}{M} \\ =& \frac{(1+b^2)(\alpha+\gamma) + 2(1-b^2)\beta}{bM} + \frac{2b\alpha}{bM} \text{ where } b = \frac{k}{1-k} \\ =& \frac{[(1+x) - b(1-x)]^T(D-W)[(1+x) - b(1-x)]}{b1^TD1} \end{aligned} \tag{2.1.6}$$

This problem can be changed to

$$min_y \frac{y^T(D-W)y}{y^TDy} \text{ with constraint } y_i \in \{1, -b\} \tag{2.1.7}$$

This is the Generalized eigenvalue problem $(D-W)y = \lambda Dy$ with constraint, but we solve it as if constraint does not exist. Since $D - W$ is a positive



(a) Proceeding of the Normalized Cut

(b) Result of the Normalized Cut

Figure 2.2: Result of the Normalized Cut Algorithm

semi-definite matrix and for $y = 1, \lambda = 0$, the original problem turns to one finding second smallest eigenvector. Since y is discrete, we turn elements of the eigenvector. Usually, using signs is enough.

If we want to divide into more than two segments, third or latter eigenvectors can be used in theory. But in practice, we will get poor results, since errors occurred in problem approximation become larger. For this reason,

if you want to produce multiple segments, you should use a bi-partitioning iterative as like in (a) in Figure 2.2. Due to this constraint along with the slow speed, it is not suitable to create a large number of superpixels although it shows decent boundary adherence.

There is another graph-based superpixel generating method, called FH methods [4], named after the writer of the paper, P. F. Felzenszwalb and D. P. Huttenlocher. FH algorithm is surprisingly fast, even though it is a graph-based method since it is based on a greedy algorithm. On the other hand, this algorithm generating extremely irregular and unbalanced superpixels as shown in Figure 2.3, and we can't control the number of the segments and the compactness of the superpixel. This is why some people don't categorize this algorithm as a superpixel algorithm. They think this is just over-segmentation.

Algorithm 1 FH Algorithm

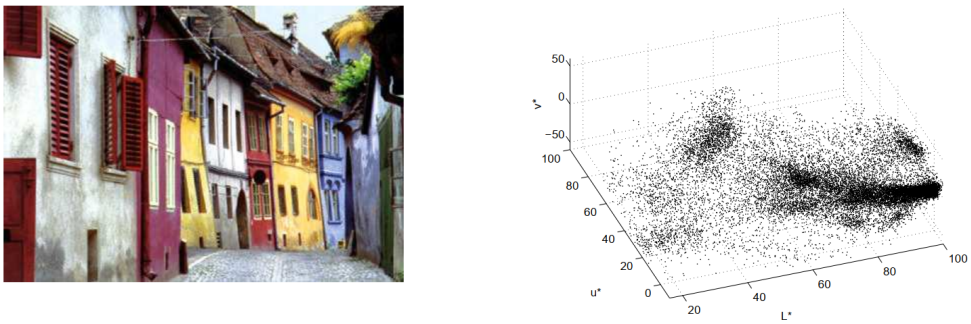
1: Convert Image into graph $G = (V, E)$.
   V is the all pixels in the image.
   Generate E by connecting nearest n points or points in some range.
   Weights of edges are proportional to similarity of two pixels.
2: Set each $v \in V$ as the cluster in graph.
3: Sort E into $= (o_1, \cdots, o_m)$, by non-decreasing edge weight
4: **for** $q = 1, \cdots m$ **do**
5: Pick o_q that connect cluster C_i and C_j
6: Calculate the minimum internal distance as below:

$$MInt(C_1, C_2) := min(Int(C_1) + \sigma(C_1), Int(C_2) + \sigma(C_2))$$

   where $Int(C) := max_{e \in MST(C,E)} w(e)$ and $\sigma(C) := \frac{k}{|C|}$.
   MST: the minimum spanning tree and $|C|$ is the size of the cluster C.

7: **if** $w(o_q) < MInt$ **then**
8: Merge two clusters.
9: **end if**
10: **end for**

Figure 2.3: Result of the FH Algorithm



(a) (b)

Figure 2.4: An image and the pixels of them [5]

Mean Shift [5] is another type of superpixel generating algorithm. This is the density-based algorithm which performs mode-seeking in a computed density image; each pixel is assigned to the corresponding mode it falls into. This algorithm adopts mean shift procedure [6]. We can make a sequence $\{y_j\}_{j=1,2,\cdots}$ from the initial centroid, y_1 by iteratively re-estimating the new centroid using the given weight kernel function $K(x)$.

$$y_{j+1} = \frac{\Sigma_{i=1}^{n} x_i K(||\frac{y_j - x_i}{h}||^2)}{\Sigma_{i=1}^{n} K(||\frac{y_j - x_i}{h}||^2)}, \qquad j = 1, 2, 3, \cdots \tag{2.1.8}$$

The mean shift procedure has been proven to converge if the kernel is good enough, so it guarantees the stability of the algorithm.

Theorem 2.1.1. ***Convergence of the Mean Shift Procedure [5]***
If the kernel K has a convex and monotonically decreasing profile, the sequences $\{y_j\}_{j=1,2,\cdots}$ converges.

We can turn images into collections of points in 5-dimensional space, which is the union of 2-dimensional space coordinates and 3-dimensional color coordinates. Figure 2.4 shows the scatter plot of the pixels in the image in the Lab color space. By applying the Mean Shift Procedure to these points, we can gain superpixels of image. Figure 2.5 is the result of the Mean shift segmentation applying to the 'cameraman.tif'. We used the Gaussian kernel as K.

Algorithm 2 Mean Shift Segmentation Algorithm

1: Run the mean shift procedure for the image.
   Each pixel x_i has its own convergence point z_i.
2: Delineate in the joint domain the clusters $\{C_p\}_{p=1,\cdots m}$ by grouping together all z_i which are closer than h_s in the spatial domain and h_r in the range domain.
3: Assign each pixel i, $L_i = \{p | z_i \in C_p\}$.

Figure 2.5: Result of the Mean Shift Algorithm

Mean Shift algorithm has some advantages. It determines its own cluster number automatically based on the bandwidth value, so highly automotive. Also, it is very robust and good boundary adherence, so many superpixel-based algorithms that don't care about speed usually adopt this method. But the biggest problem is that this algorithm is extremely slow. In chapter 3, we show the speed of the mean shift procedure in more detail. It takes minutes if we apply to ordinary-sized images. This is why it is difficult to actually use this algorithm in problems, even though it has the generality that can be applied in many fields as well as image processing.

Algorithm 3 Simple Linear Iterative Clustering Algorithm

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ using equally spaced K seeds.
2: Move the cluster centers to the lowest gradient position in a 3×3 neighborhood of seeds.
3: Set $l(i) = -1$ and $d(i) = \infty$ for each pixel i.
4: **repeat**
5: **for** each cluster center C_k **do**
6: **for** each pixel i in a $2S \times 2S$ region around C_k **do**
7: Compute the distance D between C_k and i.
8: **if** $D < d(i)$ **then**
9: set $d(i) = D$ and $l(i) = k$
10: **end if**
11: **end for**
12: **end for**
13: Compute new cluster centers
14: Compute Residual Error E
15:
16: **until** $E \leq$ threshold
17: Divide each disconnected superpixel into several connected superpixels.
18: Eliminate too small superpixels by merging most similar adjacent superpixel

Simple Linear Iterative Clustering(SLIC) algorithm [8] is one of the most popular superpixel generating methods, classified clustering-based algorithm. It has an analogous idea as the K-means clustering algorithm, updating the label of each point and position of the center of clusters alternatively. But SLIC algorithm updates only pixels nearby centers while the K-mean algorithm updates the label of all pixels globally. Therefore, it is much faster than the K-means algorithm and one of the fastest superpixel generating methods. This is why the SLIC algorithm is used in most of the superpixel applications. But this algorithm also has a major disadvantage. To guarantee the connectivity of the superpixels, post-processing is essential. In this process, the number of superpixels is changed from the value we first intended. This algorithm is not suitable in situations in which the number of superpixels must be fixed like when you're going to use it with a convolutional neural network.

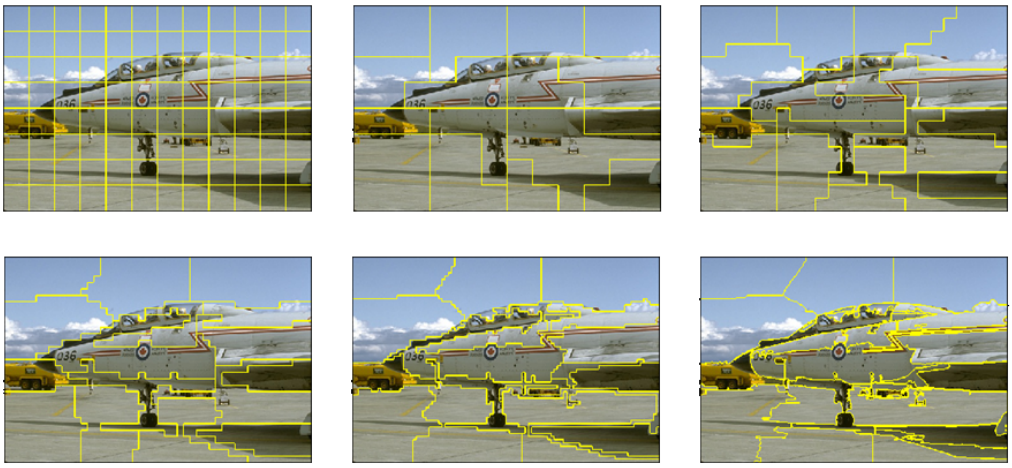


Figure 2.6: Coarse Grid to Fine Grid

ETPS [9] is another type of superpixel generating algorithm based on minimizing energy function. As shown in Figure 2.6, ETPS divides the images into the big 'blocks' of the pixels, and perform segmentation on these grids. In the segmentation process, adjacent superpixels exchange boundary blocks to minimize the energy function, greedily. When there are no more

exchanges, we call this algorithm converges in the given grid.

The energy function which we try to minimize is defined as below:

$$\begin{aligned} E_{mono}(s,\mu,c) =& \Sigma_p E_{col}(s_p, c_{s_p}) + \lambda_{pos}\Sigma_p E_{pos}(s_p, \mu_{s_p}) \\ &+ \lambda_b \Sigma_p \Sigma_{q\in N_8} E_b(s_p, s_q) + E_{topo}(s) + E_{size}(s) \end{aligned} \tag{2.1.9}$$

In the equation above, $s = (s_1, \cdots s_N)$ be the set of all random variables representing the segmentation with N the number of pixels in the image. Each $s_p \in \{1, \cdots M\}$ where M is the number of superpixels. And $\mu = (\mu_1, \cdots \mu_M)$ and $c = (c_1, \cdots c_M)$ where μ_i is the mean position of the i-th superpixel and c_i is the mean color of the i-th superpixel.

Each term of the energy function is defined as below:

$$E_{col}(s_p, c_{s_p}) = (I(p) - c_{s_p})^2$$

$$E_{pos}(s_p, \mu_{s_p}) = ||p - \mu_{s_p}||^2$$

$$E_b(s_p, s_q) = \begin{cases} 1, & \text{if } s_p \neq s_q \\ 0, & \text{otherwise} \end{cases}$$

E_{col} is the term for color homogeneity. It is the square of the difference between the color value of the pixels and the mean color of the superpixels belonging. E_{pos} is the term for shape regularity. This means the square of the distance from the pixels and the centroid of the superpixels. E_b is the term for boundary length. N_8 here means the 8 neighborhood of pixel p. If you want compact superpixels, then λ_{pos} and λ_b should have larger values. $E_{topo}(s)$ and $E_{size}(s)$ are constraint terms. If connectivity is broken, $E_{topo}(s) = \infty$. Otherwise, $E_{topo}(s) = 0$. In the same way, if the superpixels become smaller than a certain size(smaller than $\frac{1}{4}$ in original paper), $E_{size}(s) = \infty$. Otherwise, $E_{size}(s) = 0$.

Algorithm 4 Coarse to Fine Segmentation

```
Initialize superpixels with a regular grid.
Compute initial μ_i and c_i for each segment i.
for l = 1 to levelMax do.
    Initialize each block on level l with a regular grid
    Compute mean color and position in each block
    for iter = 1 to maxIters do
        Initialize list with all boundary blocks on level l
        while list is not empty do
            pop out boundary block b_i^l from list
            if connectivity(b_i^l) is valid then
                S_{b_i^l} = argmin_{s_{b_i^l}} E_mono(s, μ, c)
                if s_{b_i^l} is updated then
                    compute μ and c for the two superpixels involved.
                    append neighbors of b_i^l if they are boundary to the list.
                end if
            end if
        end while
    end for
end for
```

If this algorithm is converged on a given grid, then we make a finer grid and perform the above process again. This procedure is repeated until the boxes are reduced to single pixels.

ETPS shows excellent boundary adherence. If we judge superpixel methods only on boundary adherence, it would be the best method. Its speed is moderate, not as slow as the Ncut or the Mean shift, but 10 times slower than SLIC. One critical disadvantage of the ETPS is that if the two similar objects have a narrow gap, it tends to bind them together by forming a thought path between them.

2.1.2 Performance Comparison between the Superpixels and Choosing the Method

Stutz, Hermans, and Leibe [10] tested and compared the performance of most superpixel algorithms in their work in 2018. They ranked various superpixel generation methods and over-segmentation algorithms based on performance on five different image datasets. Their benchmark for test are Boundary Recall(Rec) [11], Undersegmentation Error(UE) [12] and Explained Variation(EV) [13]. Three metrics are defined as below:

$$Rec(G,S) = \frac{TP(G,S)}{TP(G,S) + FN(G,S)} \tag{2.1.10}$$

$$UE(G,S) = \frac{1}{|G|}\Sigma_{G_i}\Sigma_{G_i \cap S_j \neq \phi} \min\{|S_j \cap G_i|, |S_j - G_i|\} \tag{2.1.11}$$

$$EV(S) = \frac{\Sigma_{S_j}|S_j|(\mu(S_j) - \mu(I))^2}{\Sigma_{x_n}(I(x_n) - \mu(I))^2} \tag{2.1.12}$$

where $G = \{G_i\}$ is the ground truth for partition and $S = \{S_j\}_{j=1}^{K}$ is result of the superpixel segmentation. $TP(G,S)$ and $FN(G,S)$ are true positive boundary pixels and false negative boundary pixels in S with respect to G. Finally, I is the pixel value and $\mu(I)$ and $\mu(S_j)$ are mean pixel values of image and jth superpixel, respectively. Among the 28 algorithms, ETPS takes first place while SLIC, which is the most commonly used, comes in seventh place.

However, in the process of verifying the results of the experiment, we find that the performance difference between high-rank algorithms is very fine. Rather, it is much more important to increase the number of superpixel segments than adopting better algorithms, so the speed is the critical factor in applications of superpixels unless their innate properties are not fit for the problems. We adopt the SLIC algorithm as the main superpixel generating method in this paper since it is faster than most other algorithms, easy to implement and most of all it does not have any constraints. Although ETPS shows better boundary adherence, it is about 7 times slower.

2.2 Image Quality Assessment System and Superpixels

2.2.1 Object of Image Evaluation System

Fine image evaluation systems are essential in image processing in many ways. Assume there is a new denoising method. Without such systems, we judge subjectively, whether it is superior to the existing methods or not. But if we have good Image Quality Assessment(IQA) systems and huge image data set, we can determine objectively which algorithm is better. Moreover, it provides insight into the Human Visual System(HVS). If you find an IQA system that works similar to an HVS, you can assume that HVS will behave analogously to that algorithm. Thus, the search for a new frame for the IQA system is identical to an effort to better understanding the HVS.

The ultimate goal of IQA is to perfectly mimic the human cognitive system, which requires the means to measure the HVS. To gauge the quality of the images by a human, we define Mean Opinion Score(MOS) and Differential Mean Opinion Score(DMOS) as below:

Definition 2.2.1. MOS and DMOS

$$MOS = \frac{\Sigma_n^N R_n}{N} \text{ where } R_n \in \{1, 2, 3, 4, 5\}$$

$$DMOS = \frac{\Sigma_n^N D_n}{N} \text{ where } D_n \in \{1, 2, 3, 4, 5\}$$

where R_n is a score of the image given by nth person subjectively, and D_n is a score of how much a given image has been damaged compared to the original one by nth person, subjectively.

These scores differ from person to person, even though they are the same image, so more than 50 to 100 people participate and use the average values. As the definition above shows, an image with higher quality has a higher MOS and a lower DMOS.

Definition 2.2.2. Pearson's Linear Correlation Coefficient(PLCC)

In statistics, Pearson's Linear Correlation Coefficient(PLCC) is a measure

of the linear correlation between two variables X and Y defined as below.

$$PLCC = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{\Sigma_i^n (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\Sigma_i^n (X_i - \overline{X})^2}\sqrt{\Sigma_i^n (Y_i - \overline{Y})^2}} \tag{2.2.1}$$

The PLCC has values between -1 and 1. 1 means two data have a perfect positive linear correlation and -1 means they have a perfect negative correlation.

Definition 2.2.3. Spearman's Rank Order Correlation Coefficient(SROCC)

Spearman's Rank Order Correlation Coefficient(SROCC), named after Charles Spearman, is a nonparametric measure of rank correlation. This value is calculated by Pearson's Linear Correlation Coefficient of the rank variables. A ranked variable is an ordinal variable; a variable where every data point can be put in order (1st, 2nd, 3rd, etc.).

$$SROCC(X,Y) = PLCC(r_X, r_Y) = \frac{cov(r_X, r_Y)}{\sigma_{r_X}\sigma_{r_Y}} \tag{2.2.2}$$

Definition 2.2.4. Kendall's Rank Order Correlation Coefficient(KROCC)

Kendall's Rank Order Correlation Coefficient(SROCC), name after Maurice Kendall, is a statistic used to measure the ordinal association between two measured quantities. This commonly referred to as Kendall's tau also and calculated as below:

$$\tau = \frac{(\text{number of concordant pairs}) - (\text{number of discordant pairs})}{\frac{n(n-1)}{2}} \tag{2.2.3}$$

We use three metrics above as a benchmark for IQA systems. PLCC is the metric for prediction accuracy and SROCC and KROCC are the ones for prediction monotonicity.

2.2.2 Various Image Quality Assessment System

There are a lot of IQA systems and many ways to classify them. Depending on how much information of original images are provided, we can

classify IQA systems into the Full Reference(FR) systems, the Reduced Reference(RR) systems, and the No Reference(NR) systems. Here, we will only deal with the FR systems where we know all the information about the original image.

IQA systems also can be classified into two categories by its philosophy, bottom-up methods, and top-down methods. Bottom-up methods use low-level features of the original and target images. The score of these methods is calculated by the summing of local error strength.

Definition 2.2.5. MSE and PSNR

$$MSE = \frac{1}{n}\Sigma_k^n (X_k - Y_k)^2$$

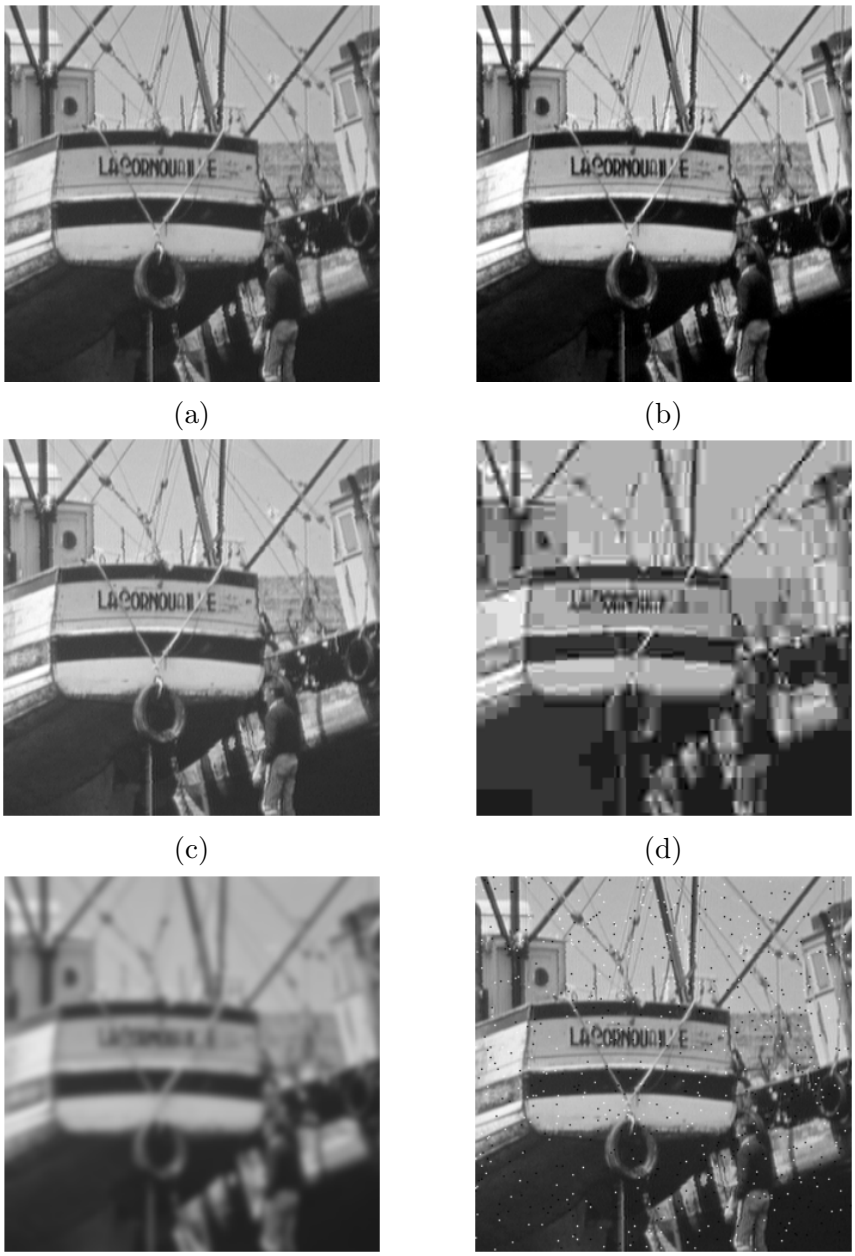
where X is an original image and Y is a distorted image, and X_k means the kth pixel of the image X.

$$PSNR = 10 \cdot log_{10}(\frac{Max(X)^2}{MSE})$$

where $Max(X)$ is the maximum pixel values in the image X.

The most popular bottom-up IQA systems are Mean Square Error(MSE) and Peak Signal-to-Noise Ratio(PSNR). Also, they may be the most commonly used IQA system, since they are easy to calculate and very intuitive for everyone. These scores are made by summing up the difference of each pair of pixels, all of the local errors.

But the problem is that MSE and PSNR are far from the human cognitive system. All of the images in the figure above have the same PSNR values, but the qualities of images, based on MOS and DMOS, are not in the same level at all. The image (b) of the figure strengthens the contrast of the original one, so all the significant information of the image has been preserved. On the other hand, in the image (e) of the figure, we can't read the letters on the back of the boat. This shows the limitations of pixel-based bottom-up IQA systems. These methods cannot distinguish between 'unimportant differences' and 'deadly differences'.



(a) (b) (c) (d) (e) (f)

Figure 2.7: Images of Same PSNR [14]

Top-down methods, in opposite categories, try to measure how well the structures of the original images are preserved in target images. To this end, various methods have been developed to measure the structure of images. Structural Similarity [14] is one of the earliest top-down methods.

Definition 2.2.6. Structural Similarity

Structural Similarity(SSIM) is the multiplication of three factors. Luminance factors are using the mean value of pixels while other factors are using the standard deviation of the pixel values.

$$\text{SSIM} = \text{Luminance Factor(l)} \times \text{Contrast Factor(c)} \times \text{Structure Factor(s)}$$

$$\textbf{SSIM} = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \cdot \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \cdot \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{2.2.4}$$

where μ is the mean of the image and σ is the std of the image. They are calculated on 8×8 windows, not globally. c_is are the small constant for stability.

Since only a local area in the image can be perceived with high resolution by the human observer, SSIM is not calculated globally. It is computed within a local square window, which moves pixel-by-pixel over the entire images and summing them up.

Theorem 2.2.7. ***Properties of the Structural Similarity***

1. *Symmetry:* $S(x, y) = S(y, x)$
2. *Boundedness:* $S(x, y) \leq 1$
3. *Unique Maximum:* $S(x, y) = 1$ *if and only if* $x = y$ *for all pixel*

Structural Similarity has many good properties. The maximum value of SSIM is 1 and it occurs only when two images are equivalent. It doesn't need normalization and it is easier to understand image quality than PSNR. Wang, himself, improved this method and suggest Mean Structural Similarity(MSSIM) [15]. Instead of local square windows, he proposed 11×11 circular symmetric Gaussian weighting function to avoid block artifacts.

Definition 2.2.8. Mean Structural Similarity

Mean Structural Similarity(MSSIM) is the advanced version of SSIM.

$$\mu_x = \Sigma_{i=1}^{N} w_i x_i$$
$$\sigma_x = (\Sigma_{i=1}^{N} w_i (x_i - \mu_x)^2)^{0.5}$$
$$\sigma_{xy} = \Sigma_i = 1^N w_i (x_i - \mu_x)(y_i - \mu_y)$$

where w is the 11×11 Circular Symmetric Gaussian weight.

$$\mathbf{SSIM}(x_j, y_j) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \cdot \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \cdot \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}$$

using values above, and

$$\mathbf{MSSIM}(X, Y) = \frac{1}{M} \Sigma_{j=1}^{M} \mathbf{SSIM}(x_j, y_j) \tag{2.2.5}$$

Wang also suggests Multi-Scale Structural Similarity(MS-SSIM) [16]. The perceivability of image details depends on the sampling density of the image signal, the distance from the image plane to the observer, and the perceptual capability of the observer's visual system. A single-scale method as described above may be appropriate only for specific settings, but it is bound to be affected by the resolution of images. so the multi-scale method is a convenient way to incorporate image details at different resolutions.

MS-SSIM creates image pyramids by taking the reference and distorted image signals as the input, the system iteratively applies a low-pass filter and downsamples the filtered image by a factor of 2. We call the original image as scale 1, and highest scale as Scale M, which is obtained after $M-1$ iterations.

Definition 2.2.9. Multi-Scale Structural Similarity

Multi-Scale Structural Similarity(MS-SSIM) is the advanced version of SSIM, using image pyramids.

$l_j(x, y)$:= luminance factor at Scale j
$c_j(x, y)$:= contrast factor at Scale j
$s_j(x, y)$:= structure factor at Scale j

Figure 2.8: Gradient Changes and SSIM [17]

$$\textbf{MS-SSIM} = l_M(x,y)^{\alpha_M} \cdot \Pi_{j=1}^{M} c_j(x,y)^{\beta_j} s_j(x,y)^{\gamma_j} \tag{2.2.6}$$

where $\alpha_j = \beta_j = \gamma_j$ and $\Sigma_{j=1}^{M}\gamma_j = 1$

What is noticeable is that the only luminance factor of the highest scale is used. For gamma value, Wang used $\gamma_1 = 0.0448, \gamma_2 = 0.2856\gamma_3 = 0.3001\gamma_4 = 0.2363, \gamma_5 = 0.1333$ for 5 scale MS-SSIM.

There are some applications based on the ideas of SSIM. The second image of Figure 2.8 is heavily damaged in the process of JPEG compression and the third image is contaminated by Gaussian noise. Despite MOS and DMOS indicates that people regard that third image has better quality than the second one, SSIM gives a higher score on the second image. This happens since SSIM does not take much into consideration the gradient. To improve

0	0	0	0	0
1	3	8	3	1
0	0	0	0	0
-1	-3	-8	-3	-1
0	0	0	0	0

M_1

0	0	1	0	0
0	8	3	0	0
1	3	0	-3	-1
0	0	-3	-8	0
0	0	-1	0	0

M_2

0	0	1	0	0
0	0	3	8	0
-1	-3	0	3	1
0	-8	-3	0	0
0	0	-1	0	0

M_3

0	1	0	-1	0
0	3	0	-3	0
0	8	0	-8	0
0	3	0	-3	0
0	1	0	-1	0

M_4

Figure 2.9: Operators for calculating the Gradient Value [17]

these shortcoming, Liu and Lin [17] proposed the Gradient similarity(GSIM).

Gradients on each pixel are calculated using operators in Figure 2.9. For pixel x, g_x is defined as below:

$$g_x = max_k \ (|x \circledast M_k|) \text{ where } k = \{1, 2, 3, 4\}$$

For pixel x in original image, and corresponding pixel y in distorted image, we define

$$R = \frac{|g_x - g_y|}{max(g_x, g_y)}$$

Using R, we can calculate the gradient similarity at pixel x and y. And Gradient Similarity of image is defined by weighted sum of them,

$$g(x, y) = \frac{2(1 - R) + K}{1 + (1 - R)^2 + K} \text{ where } K = \frac{10^{-5}}{\max(g_x, g_y)}$$

$$GSIM = (1 - W(g, e)) \cdot g + W(g, e) \cdot e \text{ where } W(g, e) = c \cdot g$$

Weight is proportional to the gradient value, because the larger the gradient, the more likely it is to contain critical information such as edges.

2.2.3 Applying Superpixels to IQA System

(a) SPSIM Patch (b) SSIM Patch

Figure 2.10: Patches of the SSIM and SPSIM [18]

Many IQA systems adopt image patches for local context information. But in most cases, these patches do not connote any visual meanings. Original SSIM and many other IQA systems using square patches without pro-

viding any apparent grounds for that. Even Wang [14] pointed out that this frame may occur unnecessary artifacts.

Images (b) in the figure 2.10 shows the problem of the square frame. The square patch in the image has two disparate regions simultaneously. The top of the patch is relatively monotonous, but the bottom part consists of a red pattern on a white background. It is very challenging to gauge errors on these frames. It is much desirable that only the area of the same properties exists in a single frame, like (a) in figure 2.10.

Superpixel may be an alternative to creating patches that meet these conditions. Sum et al. [18] proposed a new IQA system call Superpixel-based Similarity index(SPSIM) using superpixels instead of square patches. SPSIM calculates similarity factors on YUV color space [19] while original SSIM uses RGB color space.

$$L_i = \frac{1}{|s_i|}\Sigma_{j\in s_i}Y(j), \qquad M_L(i) = \frac{2L_r(i)L_d(i)+T_1}{L_r^2(i)+L_d^2(i)+T_1} \tag{2.2.7}$$

where s_i means the superpixel containing ith pixel, and $Y(j)$ is the luminance factor of the pixel j. Superpixels are generated only on original reference images and these frames are applied to both reference images and distorted images. SLIC algorithm is adopted to generate superpixels. $L_r(i)$ and $L_d(i)$ are mean Luminance value of the superpixel s_i in reference images and distorted images, respectively. Superpixelwise luminance similarities(M_L) are define as above.

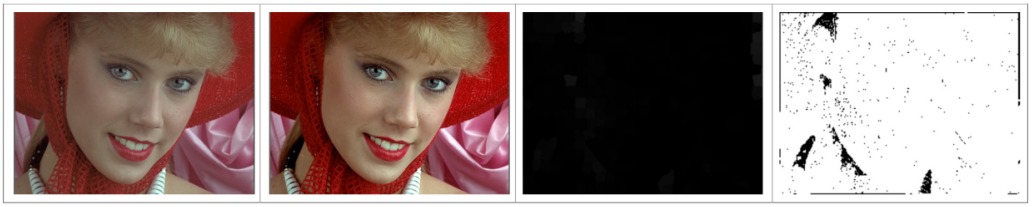
$$U_i = \frac{1}{|s_i|}\Sigma_{j\in s_i}U(j), \qquad M_U(i) = \frac{2U_r(i)U_d(i)+T_1}{U_r^2(i)+U_d^2(i)+T_1} \tag{2.2.8}$$

$$V_i = \frac{1}{|s_i|}\Sigma_{j\in s_i}V(j), \qquad M_V(i) = \frac{2V_r(i)V_d(i)+T_1}{V_r^2(i)+V_d^2(i)+T_1} \tag{2.2.9}$$

Similarly, we can compute similarities of U and V elements. Chrominance similarity is defined as

$$M_C(i) = M_U(i)M_V(i)$$

SPSIM also focuses on gradients of each pixel in images. Vertical and



(a) MOS = 7.2143, GSIM = 0.9735

(b) MOS=4.9286, GSIM=0.9711

(c) MOS=5.0714, GSIM=0.9774

Figure 2.11: Contrast Errors and Other Errors [18]

horizontal gradients on each pixel i, $G_v(i)$ and $G_h(i)$ respectively, are calculated using Prewitt operator. Gradient magnitude on pixel i is defined as $G(i) = \sqrt{G_v^2(i) + G_h^2(i)}$ and Gradient Similarity are defined as

$$M_G(i) = \frac{2G_r(i)G_d(i) + T_2}{G_r^2(i) + G_d^2(i) + T_2}$$

Next, SPSIM pays attention to a contrast error. Many experiments show that people do not regard the image with contrast error that has been seriously damaged. MOS of the image with contrast error is much higher compared to other types of errors with the same error power. Images in Figure 2.11 have similar GSIM values, while MOS of the images indicates that image (a) with contrast error has superior quality than the image (b), the image

with Gaussian noise and image (c), the image damaged in JPEG compression process. A similar phenomenon occurred in the Figure 2.7 before. Since not only the methods based on the pixel value but also the methods based on gradient do not measure the effect of contrast error satisfactorily. Therefore, the SPSIM algorithm tries to make the further correction for the contrast type errors. In the previous equation for luminance and chrominance similarities, T_1 and T_2 are used without proper definition. These values are used to calibrate the SPSIM score for contrast error.

To handle the contrast errors, we should distinguish it from others. SPSIM uses Regional Gradient Consistency(RGC) to judge whether distorted images have the contrast type error. This value is calculated using Spearman's Rank Order Correlation Coefficients(SROCC), defined before, of gradients in each superpixel.

Definition 2.2.10. Regional Gradient Consistency

$$RGC(S_r, S_d) = SROCC(g_r, g_d) = 1 - \frac{6\Sigma_{i=1}^{K} d_i^2}{K(K^2-1)} \tag{2.2.10}$$

The Regional Gradient Consistency(RGC) measures the consistency of the direction of errors. If the magnitude of the RGC is high, we can assume that most errors occurred in a distorted image are contrast errors. But RGC only detects the existence of the contrast error. It does not tell whether it's positive or negative. To know the direction of the contrast, SPSIM suggested the Increase or Decrease of Gradients(IDG).

Definition 2.2.11. Increase of Decrease of Gradients

$$IDG(g_r, g_d) = \frac{1}{K}\Sigma_{i=1}^{K} sign(g_d(i) - g_r(i)) \tag{2.2.11}$$

If IDG value is high, then distorted image has increased gradient.

If contrast error occurs, the SPSIM score should be raised. Therefore, if the RGC value is high, the magnitude of T_1 and T_2 value should be large. Also, positive IDG implies that gradients of images are increased, so T_1 and

T_2 should be larger than when IDG is negative. Using RGC and IDG above, SPSIM defined T_1 and T_2 as followed:

$$u_0 = psgn(RGC(S_r, S_d) - \tau_0)$$
$$u_1 = psgn(IDG(S_r, S_d) - \tau_1), \quad u_2 = psgn(-\tau_1 - IDG(g_r, g_d))$$
$$IF_A(S_r, S_d) = u_0 u_1, \qquad IF_B(S_r, S_d) = u_0 u_2$$
$$T_1(S_r, S_d) = C_1 + \lambda_1 IF_A + \lambda_2 IF_B, \qquad T_2(S_r, S_d) = C_2 + \lambda_1 IF_A + \lambda_2 IF_B$$

Since humans prefer the image with an increased gradient rather than a decreased gradient, λ_1 should be much larger than λ_2. Sun [18] got best result in his paper when $C_1 = 600, C_2 = 210, \lambda_1 = 40000$ and $\lambda_2 = 950$. So the overall comparison is expressed as:

$$M(i) = M_G(i)[M_L(i)]^{\alpha} e^{\beta(M_C(i)-1)} \tag{2.2.12}$$

where α and β are parameters to adjust the weights of luminance and chrominance similarities. SPSIM is weighted sum of $M(i)$ and weight is calculated using a standard deviation and $Kurt[X] = E[(\frac{X-\mu}{\sigma})^4]$ of the superpixels.

$$TC(i) = \frac{std(S(i))}{Kurt[S(i)] + 3}$$
$$w(i) = exp(0.05 \cdot abs(TC_d(i) - TC_r(i))$$
$$SPSIM = \frac{\Sigma_{i=1}^{N} M(i) w(i)}{\Sigma_{i=1}^{N} w(i)} \tag{2.2.13}$$

2.2.4 Performance Comparison of IQA System

Sun [18] tested various IQA systems on four databases. Data base used in experiments are LIVE [20], CSIQ [21], TID2008 [22] and TID2013 [23]. This table shows the value of the superpixel. Multi-Scale Structural Similarity is one of the most powerful image quality assessment systems, although it was developed more than a decade ago and has a relatively simple structure. By adopting the superpixel frame for image interpretation and adding a small

modification, SPSIM excels the MS-SSIM in every database. This shows that superpixel is very similar to the human cognitive system and suitable as a framework for image processing. From the next chapter, we propose new image processing methods, based on superpixels.

Table 2.1: Performance Comparison of FR IQA Methods on Data Bases [18]

		PSNR	SSIM	MS-SSIM	SPSIM
LIVE	SROCC	0.8756	0.9479	0.9513	**0.9620**
	KROCC	0.6865	0.7963	0.8045	**0.8271**
	PLCC	0.8723	0.9449	0.9489	**0.9599**
CSIQ	SROCC	0.8005	0.8756	0.9133	**0.9440**
	KROCC	0.5984	0.6907	0.7393	**0.7880**
	PLCC	0.7998	0.8613	0.8991	**0.9344**
TID2008	SROCC	0.5245	0.7749	0.8542	**0.9104**
	KROCC	0.3696	0.5768	0.6568	**0.7303**
	PLCC	0.5309	0.7732	0.8451	**0.8927**
TID2013	SROCC	0.6394	0.7417	0.7859	**0.9044**
	KROCC	0.4696	0.5588	0.6047	**0.7251**
	PLCC	0.7017	0.7895	0.8329	**0.9091**

Chapter 3

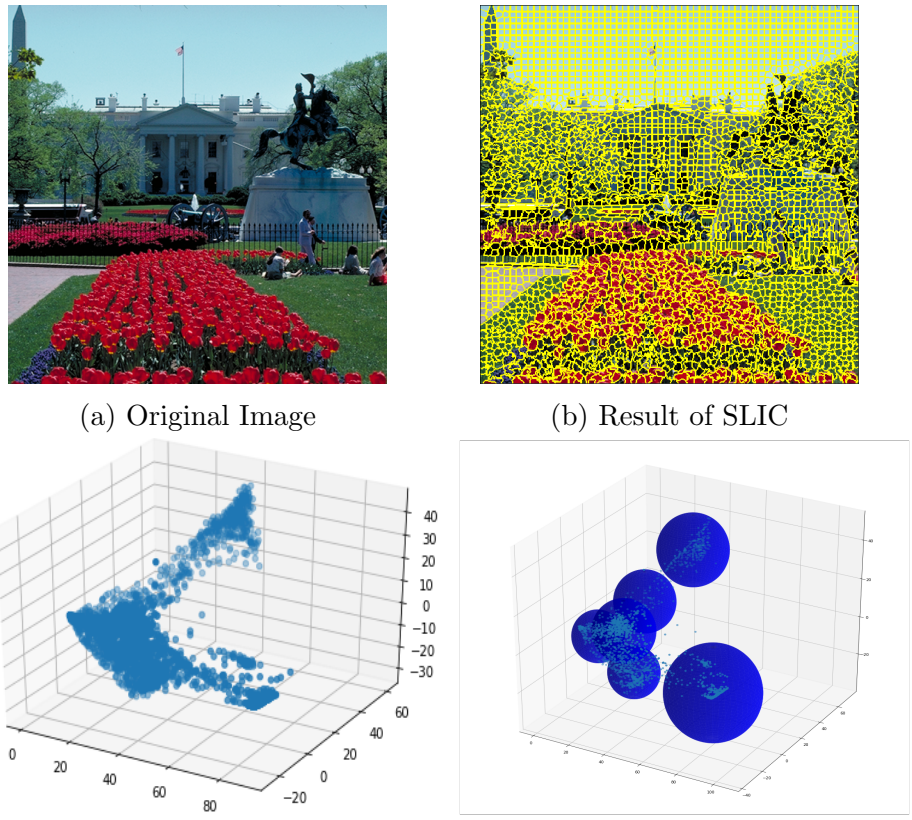
Adaptive Image Segmentation Based on Superpixel

When you observe people segment images manually, you will easily notice that the image segmentation criteria are quite inconsistent. Even with Lab color space, which best reflects human visual distances, sometimes humans recognize the more dissimilar part as one segment while dividing the more similar part. This is why it is not possible to have a consistent threshold for image segmentation. Moreover, this criterion does not differ from image to image, but even within the same image. Therefore, in order to perform human-like segmentation in complex images, an image segmentation algorithm should be highly adaptive, and an approach based on threshold has limitations in carrying out this.

Grasping the overall color structure of the image should be preceded to adaptively process the image. In the analysis of overall structure of image, one may find the pivot colors which dominate the images. This colors determines overall impression of the images and have a great influence to the segmentation. We call these values as 'Representative Color'(RC) of the image.

In this chapter, we propose a novel method called Superpixelwise Mean Shift(S-Mean Shift) combining superpixels and means shift procedure to find the RC. Next, we propose a novel image segmentation algorithm with a few parameters needed based on RC in order to overcome the limitation of threshold-based image segmentation.

3.1 Superpixelwise Mean Shift



(a) Original Image (b) Result of SLIC

(c) Mean of the Superpixels (d) Overlapping Sphere

Figure 3.1: Image and the Lab

In order for the algorithms to be 'adaptive', it is absolutely necessary to identify the color structure of the image. But if we try to figure out the structure based on the pixel information, it is too time-consuming because there are too many elements we should handle. As mentioned before, superpixel is optimized to deal with these problems. Not only does it reduce the number of elements, but it also eliminates small noises or textures, that hardly have any meaning, in this process automatically.

We apply the Simple Linear Iterative Clustering(SLIC) method to the image in the Lab color space and grouping nearby similar pixels. For each

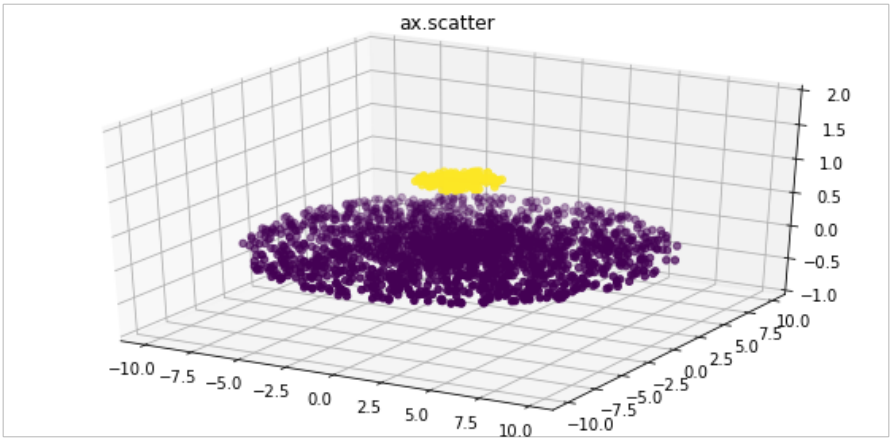


Figure 3.2: Two Thin Plate

superpixel, their average Lab color values are calculated. (c) of Figure 3.1 is the scatter plot of the average Lab color values of the superpixels. In the process of Figure 3.1, we can reduce the values we should handle to nearly 1%.

After simplifying the problem using superpixels, we tried to find RCs of the images. Previous, we try to figure out these values based on K-mean clustering. If we find the centers of K-means clustering, using the appropriate number of centers, it is assumed that these centers will be representative colors. At first, we applied existing methods, such as elbow method, average silhouette method and gap statistic method [25], to find the appropriate number of clusters of K-means clustering, but we found that all these methods are not appropriate for highly complicated data. To solve this problem, at the conference of the KSIAM 2019, we proposed a novel method.

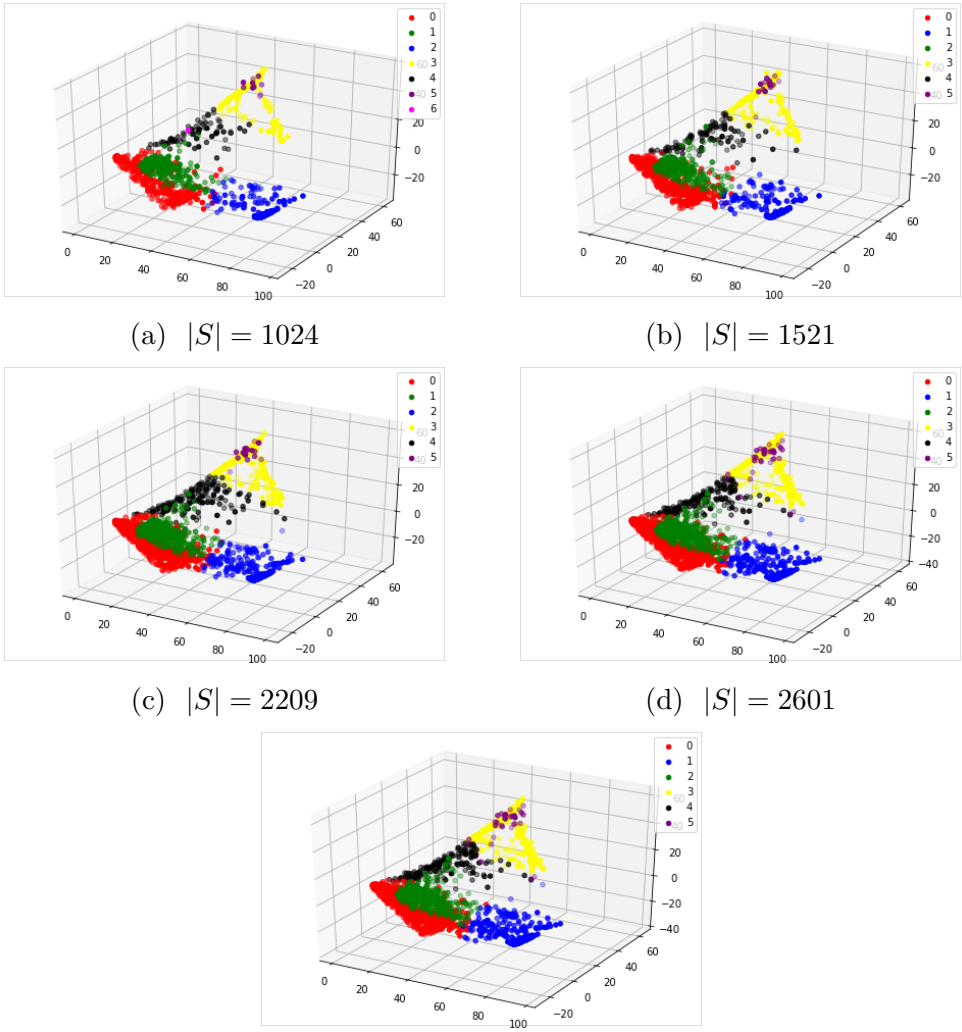
Perform K-means clustering using n centers and get $C = \{c_1, c_2, \cdots c_n\}$. For each c_i, calculate the $d_i = \min_{j \neq i} dist(c_i, c_j)$. And sort C with the d_i ascending order, make $C' = \{c_{j_1}, c_{j_2}, \cdots, c_{j_n}\}$. Set $r_{j_1} = \frac{d_{j_1}}{2}$ and $r_{j_k} = d_{j_k} - r_{j_{k-1}}$ for $k \neq 1$. Draw n spheres centered at c_i and radius r_i and check the number of points that are not in any of these spheres. By varying the n from 3 to 10, we find n which has the fewest remaining points. The reason why we set 10 as the maximum number of clusters is while testing more

than 100 images, the number of representative color is never more than 10. The graph (d) in Figure 3.1 shows the result of the algorithm. But this method has several problems. First, it is not easy to find the appropriate number of cluster centers. Most of all, it is not robust and sometimes fails since the shape of the cluster can be far from the sphere or ellipsoid. This method cannot separate the two narrowly apart plates that look as shown in Figure 3.2. To deal with various types of point collections, we need better alternatives. So we revisit the idea of the mean shift procedure.

As mentioned in the previous chapter, the mean shift procedure is not widely used due to its slow speed. On the other hand, its performance is not bad and it can handle the data of various forms. Unlike our previous method, it can divide the points in Figure 3.2. And since we can greatly reduce the number of elements by applying superpixel algorithms to an image, we can use the mean shift procedure despite its speed. Combining the advantages of the superpixel algorithm and mean shift procedure, we propose a novel method called Superpixelwise Mean Shift(S-Mean Shift) here.

Besides being able to handle various forms of data, S-Mean Shift has another advantage: it is very robust. Figure 3.3 shows the number of superpixels and the result of of the S-Mean Shift clustering. Results of the previously applied methods like K-means algorithm often vary significantly when we change the number of superpixels. On the other hand, despite the change of the number of the superpixel, it can be seen that the results of S-Mean Shift are very consistent except for the addition of a tiny cluster. For this reason, S-Mean Shift does not require many points. To achieve good results reliably in image segmentation or boundary detection, at least 3,000 superpixels are needed for 512×512 images while about 1,500 superpixels are sufficient for S-Mean Shift as shown in results. More details are on Table 3.1. All the images are selected from the CSIQ [21] dataset. $|S|$ in the column of the table means the number of the superpixels. This value changes during generation, so real $|S|$ value is different from inputted $|S|$ value. The most right columns of the tables show the number of superpixels that belong to ith cluster. Since the size of the superpixels is not the even, the number of superpixels in the clusters is not perfectly proportional to the number of the overall superpixels. But as in Figure 3.3 shows, it can be seen that the

overall trend is maintained regardless of the number of superpixels.



(a) $|S| = 1024$ (b) $|S| = 1521$

(c) $|S| = 2209$ (d) $|S| = 2601$

(e) $|S| = 3249$

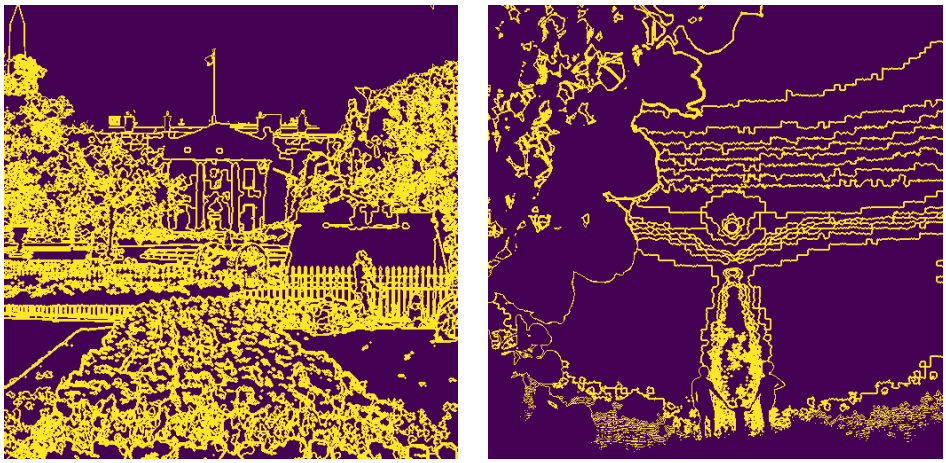
Figure 3.3: Result of Mean Shift and the Number of Superpixels(1)

Table 3.1: Result of Mean Shift on Superpixels(2)

Image Name	$\|S\|$(Setting)	$\|S\|$(Real)	0	1	2	3	4	5	6+
1600.png	1000	1024	328	269	212	130	74	10	1
	1500	1521	494	407	309	192	105	14	
	2000	2209	714	574	463	280	160	18	
	2500	2601	866	682	531	321	170	31	
	3000	3249	1073	847	672	394	206	57	
family.png	1000	1020	349	300	168	82	65	57	3
	1500	1515	521	452	237	168	140	2	1
	2000	2197	737	685	346	233	205	2	1
	2500	2592	889	791	428	255	233	4	1
	3000	3249	1073	847	672	394	206	57	
woman.png	1000	1024	209	292	204	162	56	52	49
	1500	1521	448	322	272	261	93	76	49
	2000	2209	873	544	480	120	102	90	
	2500	2601	1097	631	453	159	105	156	
	3000	3249	1308	750	749	192	131	119	
bridge.png	1000	1024	315	273	190	139	107		
	1500	1521	450	429	281	210	151		
	2000	2209	596	589	444	312	268		
	2500	2601	821	687	512	351	230		
	3000	3249	903	876	695	447	328		

3.2 Two-Step Approach using S-Mean Shift

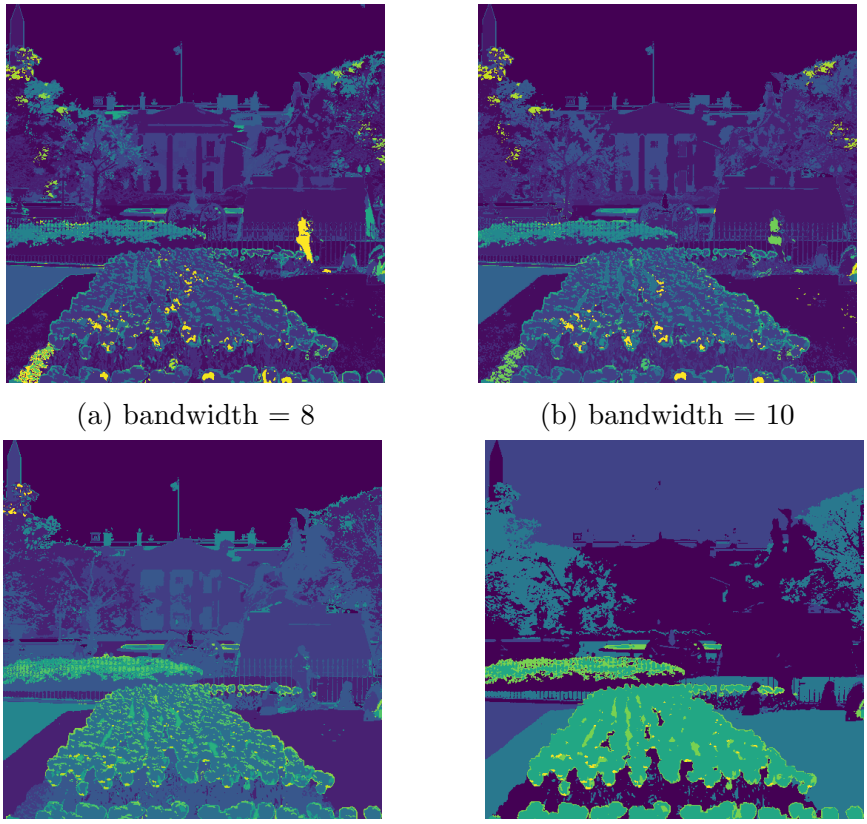
As mentioned earlier, more than 3,000 superpixels are needed to obtain a decent segmentation result from the 512 × 512 image. But the mean shift algorithm is extremely slow, therefore we can't apply it to the more than 3,000 elements since sometimes it takes more than 10 seconds. On the other hand, the SLIC algorithm is proved to be fast, and in the previous section, we show the robustness of the S-Mean Shift. So our segmentation strategy adopts a two-step approach.



(a) 1600.png (b) family.png

Figure 3.4: First Result of S-Mean shift

For reference data, we have created about 1,500 superpixels and call them as reference superpixels. Perform mean shift procedure on reference superpixels to generate reference data for the segmentation. Next, we perform second superpixel generations with the much higher number of the superpixel and call it segment superpixels. This number depends on the properties of images. If there are thin objects we should detect, more superpixels are needed. In this process, connectivity need not be guaranteed since we are conducting region labeling. S-Mean shift performance is better when it did not guarantee connectivity for most images in our experiment. Then predict the cluster of each segment superpixel based on reference data and combine



(a) bandwidth = 8

(b) bandwidth = 10

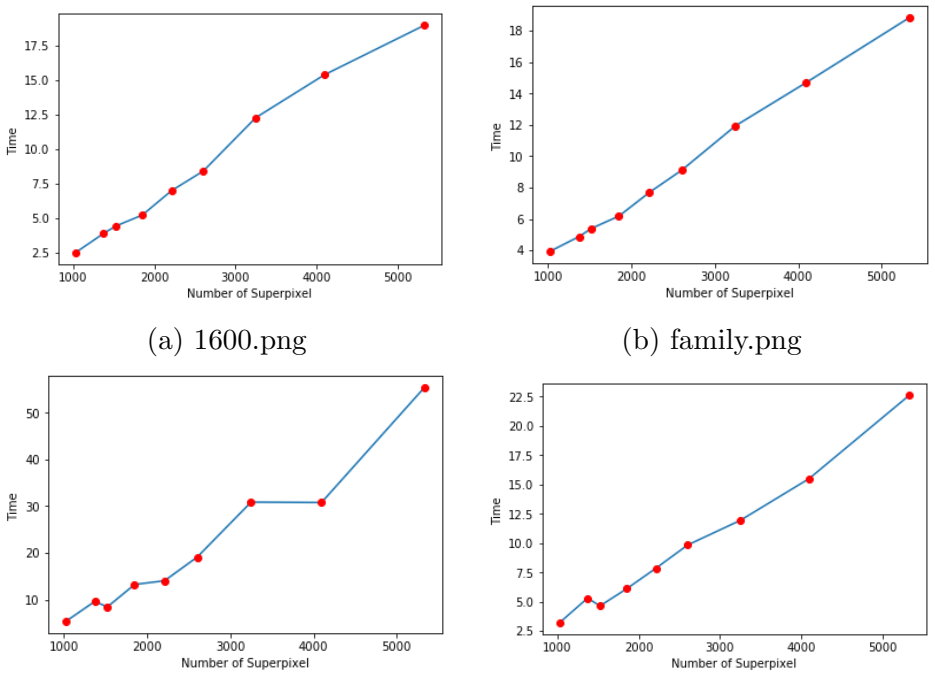
(c) bandwidth = 15

(d) bandwidth = 20

Figure 3.5: S-Mean Shift and Bandwidth

the superpixels of the same cluster. This is our first result of the S-Mean Shift and shown in Figure 3.4.

In the process of creating reference data, a bandwidth value of the mean shift procedure is the only but crucial parameter of the algorithm. In our experiments, 12 or 13 showed the best results in most images. If you need to process a huge dataset using a single parameter, a lower value would be better. This is shown in Figure 3.5. Our best choice for the bandwidth for the image '1600.png' is 13. When we set bandwidth value is higher than the desired, we will lose crucial information about the image. In (d) we lose flag pole in the center and in (c) we lose a person in front of the statue. On the other hand, when the bandwidth value is lower than desired, the effect is not



(a) 1600.png (b) family.png

(c) woman.png (d) bridge.png

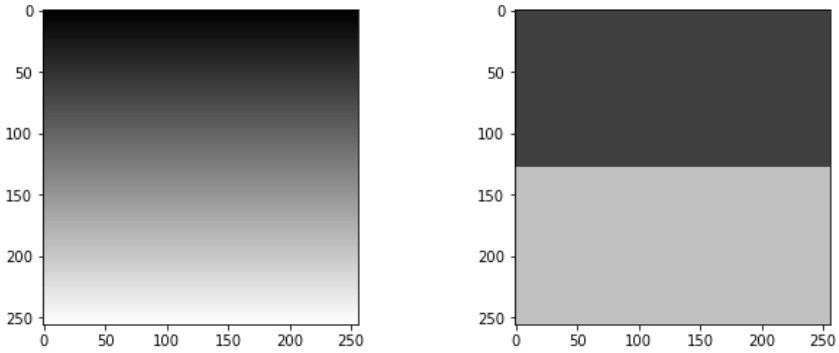
Figure 3.6: Computational Time of First Step

noticeable compared to when it is too high.

The first step of this process takes about 5∼6 seconds and the second step takes about 1∼2 seconds in the 512× 512 sized CSIQ images which are extremely complicated. Despite using fewer elements, the first step is still consuming more time than the second step. More information about the time spent in the mean shift procedure is provided in Figure 3.6. The time spent on mean shift procedure increases almost linearly with the number of elements used. Our computational environment is an ordinary personal computer without GPU and this code is implemented with python. Although post-processing is necessary, it is enormously fast compared to existing methods and already the approximated boundaries of the main objects are detected at this stage and can be applied for various purposes.

3.3 Gradient Transition and Eliminating Small Pieces

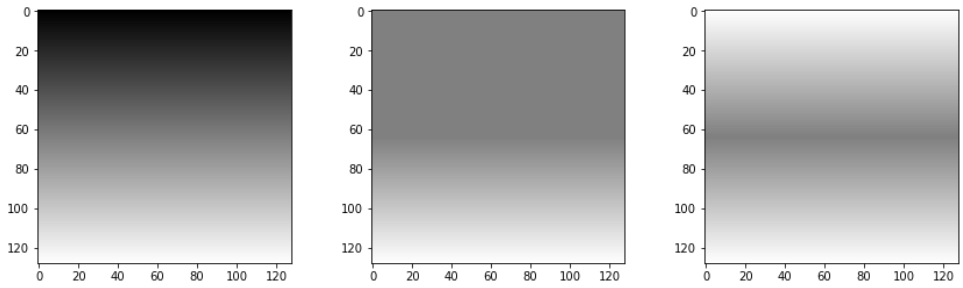
S-Mean Shift has an advantage in getting the key information from the pictures, but it still over-segment the images and generates unnecessary boundaries. In this stage, unlike the first segmentation stage, we focus on the local information. Based on the local information, we will judge the validity of the boundaries generated by the S-Mean shift method and merge the segments across the invalid borders.



(a) (b)

Figure 3.7: Gradient and Boundary

Empirically, there should be a strong contrast between both sides of the segment line to be a good segmentation. The Figure 3.7 is the toy example showing this observation. (a) of the figure is the 256×256 gray-scale image such that the intensity of pixel $I(x, y) = x$, and (b) is composed of two pieces, upper piece whose intensity is 192 and lower piece whose intensities is 64. Each upper segment of (a) and (b) has the same average intensity, 192, and each lower segment of (a) and (b) also has the same average intensity, 64, but their impression is very different. We can find the 'natural boundary' between upper and lower parts, but most people regard (a) as one bigger segment. This shows that the average value of each segment is not sufficient to find a valid border. In this post-processing, we will utilize gradient information around the segment line founded by the S-Mean shift.



(a) (b) (c)

Figure 3.8: Gradient Change and Boundary

Here's another toy example. Three images in Figure 3.8 have same lower part such as $I(x, y) = 2x$, but their upper part are different. The intensity of upper part in (a) is $I(x, y) = 2x$, in (b) is $I(x, y) = 128$ and in (c) $I(x, y) = 255 - 2x$. Most people find a stronger border in the middle of the images as they proceed to the right. This gives us another insight into the boundaries. People are more sensitive to changes in gradient than to the magnitude of the gradient, itself. Also, earlier in chapter 2, IQA systems using gradient changes shows better performance.

We hereby define a gradient transition of two segments.

Definition 3.3.1. Gradient Transition

Assume p is pixel in the segment s_1 and adjacent to s_2 and define

$$p_{s_1} := \{q \in s_1 | q \text{ is adjacent to } p\}$$

$$p_{s_2} := \{q \in s_2 | q \text{ is adjacent to } p\}$$

Then Gradient Transition from segment s_1 to segment s_2 is defined as

$$G_T(s_1, s_2) = \frac{\Sigma_{p \in b(s_1,s_2) \cup b(s_2,s_1)} \Sigma_{q_1 \in p_{s_1}} \Sigma_{q_2 \in p_{s_2}} ||(I(p) - I(q_1)) - (I(q_2) - I(p))||_2}{\Sigma_{p \in b(s_1,s_2) \cup b(s_2,s_1)} \Sigma_{q_1 \in p_{s_1}} \Sigma_{q_2 \in p_{s_2}} 1} \tag{3.3.1}$$

where $b(s_1, s_2)$ is the set of pixels in the segment s_1 and adjacent to s_2 and $I(p)$ is the Lab value of the pixel p.

The set p_{s_2} is not empty by definition and the set p_{s_1} is not empty unless

$|s_1| = 1$. So gradient transition is well defined for all adjacent segments s_1 and s_2 with $|s_1| > 1$ and $|s_2| > 1$. Here is some examples. Assume that a pixel p is in the segment s_1 and adjacent to the segment s_2. If p is not on the border of the image itself, then it has four neighboring pixels. If two of them are in s_1 and others are in s_2, then it has two inward gradient, G_{i1}, G_{i2} and two outward gradient G_{o1}, G_{o2}. In this case, we can add four elements $|G_{i1} - G_{o1}|, |G_{i1} - G_{o2}|, |G_{i2} - G_{o1}|$ and $|G_{i2} - G_{o2}|$ to the gradient transition collection from s_1 to s_2, $S_{GT}(s_1, s_2)$. Assume another case. If p is in segment s_1, having two neighbors which are in s_1 and adjacent to two other superpixels s_2, s_3. Then we can add two elements to the $S_{GT}(s_1, s_2)$ and $S_{GT}(s_1, s_3)$, respectively. Perform this process on all boundary pixels in the image. We regard $S_{GT}(s_1, s_2)$ and $S_{GT}(s_2, s_1)$ as one set, so gradient transitions from s_2 to s_1 are also added to $S_{GT}(s_1, s_2)$. Gradient transition from s_1 to s_2, $G_T(s_1, s_2)$ is the average of the elements in $S_{GT}(s_1, s_2)$. If s_1 and s_2 is not adjacent each other, we regard $G_T(s_1, s_2) = \infty$.

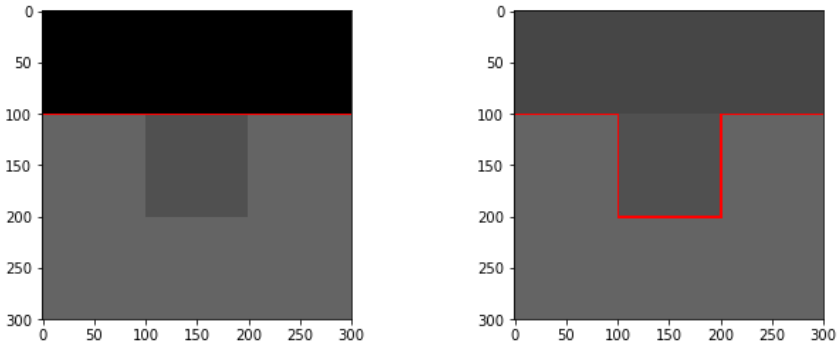
Based on the gradient transition, we will eliminate small meaningless pieces generated by S-Mean Shift. We view the images in the frame of superpixels, so superpixels are the least semantic unit in this segmentation. But, since we do not enforce connectivity in the superpixel generating process to increase boundary adherence, there are too many segments smaller than ordinary superpixels. These little pieces are too small to contain any implications. Comparing these results to manual segmentation, these results also unnatural since a person ignores small details of large images, so we should merge these things and make bigger segments. So we eliminate all pieces smaller than expected superpixel size, e-size $= \frac{\text{(Number of pixels in Image)}}{\text{(Number of Segment Superpixel)}}$. If you want to preserve smaller pieces, we should use a bigger segment superpixel number and it makes sense.

Merging small pieces is a time-consuming process since there are so many such pieces. We need to check their neighboring relation and gradient transition is not even defined for some of them. In order to perform the above task quickly, we first fill the holes of big segments which is smaller than the e-size above to reduce the computations for gradient transition. Next, we get rid of all one-pixel segments that we cannot calculate the gradient transition by merging the most similar neighboring pixel. Now, we get all the gradient

transitions for all the remaining segments.

After generating gradient transition, we process the pieces smaller than e-size and is in contact with more than two segments. These segments should not be remained alone and should be merged with the most similar neighbors. Find the most similar neighbor s_j by $j = \text{argmin}_k G_T(s_i, s_k)$ for all small segments s_i. After merging s_i and s_j, we should update the gradient transition information. $S_{GT}(s_i, s_k)$ and $S_{GT}(s_j, s_k)$ are merged, and $G_T(s_i, s_k) = G_T(s_k, s_i)$ are recalculated based on new $S_{GT}(s_i, s_k)$. Repeat this process until no small segments are remained.

3.4 Merging On Balanced Gradient Transition



(a) (b)

Figure 3.9: Influence of the other boundaries

After removing the small, meaningless pieces, we must judge the validity of the remaining borderline. If the segment line is invalid, we should merge two segments across that line. In the merging process, we divide the segments into two types based on their size.

We hereby propose the new idea that the other side of the boundaries also influence the validity of the borderline for a small segment. Figure 3.9 illustrates this phenomenon. Although the center and the lower part of the

(a) and (b) of the same intensities, the validity of the border is different since a stronger gradient transition of the other side affects the result. Based on this observation, we propose a 'Balanced Gradient Transition Criterion', which means that if the gradient transitions of the small segments should be balanced. That is, if segment s_1 is not so large and $G_T(s_1, s_2) \ll G_T(s,s_3)$ for two adjacent segment s_2 and s_3, s_1 and s_2 should be merged. If the segments are large enough, we don't apply this criterion since it's hard to see both sides at the same time.

We first process borderline with extremely low gradient transition. If the gradient transition is lower than 8, the boundary is not notable for most people. Next, we process the segments smaller 1% of the images as 'small segment' as indicated above. In the our experiment, for each small segment s, if $\min_{s_1 \in A(s)} G_T(s, s_1) < 0.25 \cdot \max_{s_1 \in A(s)} G_T(s, s_1)$ where $A(s)$ is the all segments adjacent to s, we merge s and $\text{argmin}_{s_1 \in A(s)} G_T(s, s_1)$. After merging, we update the gradient transition and the size of the segments and repeat this process until no more merging occurs.

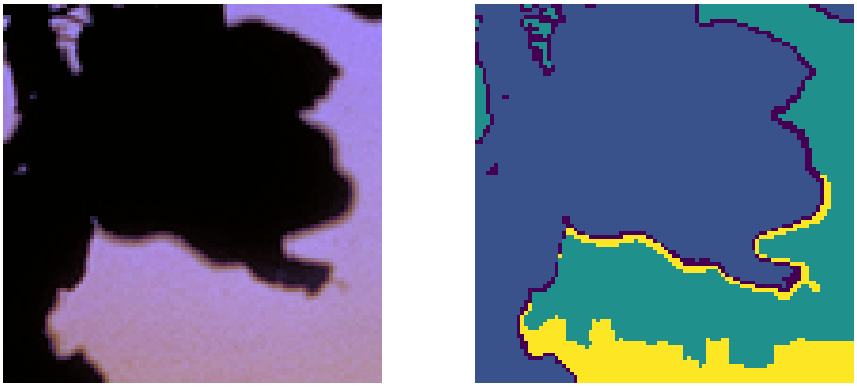
For the bigger segments s_1, we adopt the mean gradient transition of the segments.

$$\text{Mean Gradient Transition}(s_1) = \frac{\Sigma_{s_2 \in A(s_1)} |S_{GT}(s_1, s_2)| G_T(s_1, s_2)}{\Sigma_{s_2 \in A(s_1)} |S_{GT}(s_1, s_2)|}$$

If the $G_T(s_1, s_2)$ are smaller than half of the mean gradient transition of the s_1, s_1 and s_2 should be merged. Using a maximum G_T is not good for big segments since a small peculiar segment may greatly affect the segmentation result around the big segments. As above we iterate this process until no more merging happens.

Last, we check the small segments surrounded by one part. Since manual segmentation does not prefer such isolated partitions, they must pass stricter conditions to be validated. If such a segment is surrounded by a small segment, its gradient transition is more than half of the maximum gradient transition of the small segment not to be merged. Also, if such a segment is surrounded by a big segment, its gradient transition should be larger than the mean gradient transition of the big segment to be validated.

The post-processing that we should carry out is the removal of thin lay-



(a) Original Image (b) Segmentation Result

Figure 3.10: Thin Layers and Segmentation

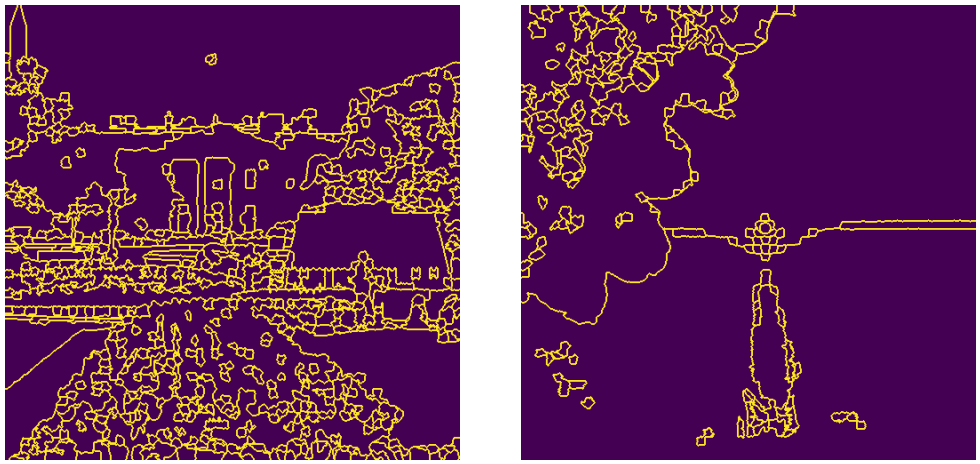
ers. As the image (a) in Figure 3.10 shows, it is quite common for the edges of an object to become blurred or have double outlines in the photograph. So if we segment these images based on color information, segmented results have double outlines. But people never made such segmentation since these thin layers have no implications. Moreover, since people prefer a compact segmentation, they don't make a thin line segment, even if they are not double outlines. To detect these segments, we compare the size of the segments and the length of its borderline.

After removing thin layers, we go up to the small segment removing process. If no changes occur in all the processes, 'merging small', 'merging big', 'merging isolated' and 'thin layer removal', our algorithm terminates. In most cases, our results hardly changed after the second loop. So if time is a more important issue, this algorithm may be ended in one loop.

3.5 Experimental Result

3.5.1 Experiment of CSIQ Dataset

CSIQ dataset [21] is the dataset for the image quality assessment system. This dataset includes 30 original images and each original image has 30 distorted versions of 6 types and 4 or 5 levels. The distortions used in CSIQ are JPEG compression, JPEG-2000 compression, global contrast decrements, additive pink Gaussian noise, and Gaussian blurring. In total, there are 866 distorted images.



(a) 1600.png (b) family.png

Figure 3.11: Threshold based segmentation for CSIQ images

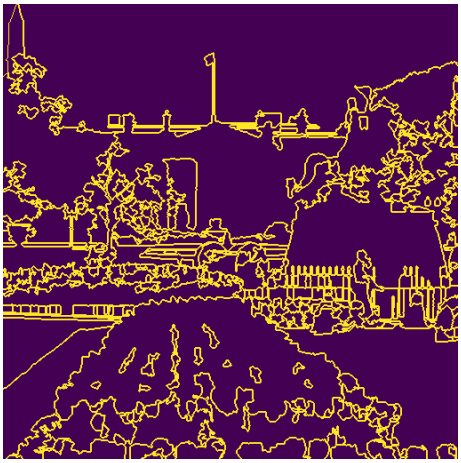
This dataset is extremely challenging for image segmentation. The images in the dataset have many objects and complicated backgrounds. Some images have difficulties in acquiring consistent results even to be segmentation manually. If we apply segmentation algorithms based on a threshold, we will get poor results. Figure 3.11 shows the segmentation result using SLIC superpixels and regions adjacency graph(RAG) [26] method. This is one of the standard methods proposed by the scikit-image package in python, but their result is bad. We lose the pole of the flag while rosebuds are oversegmented in the first image. And in the second image, we lose families which are the main object of the image while oversegmenting suns and its ring.

This happens since their Lab distance are higher, and we cannot overcome this problem if we use threshold based methods.

As shown in Figure 3.12 and Figure 3.13, our algorithm catches salient boundaries in the extremely complex images. Since some boundary line of the objects are faint and our algorithm does not use semantic information of the target images, we cannot catch all the boundaries of objects. But our result is much better than the result above since our method is highly adaptive and similar to the human system of cognizance. Our algorithm finds that a faint pole of the flag is more important than alternations of the rosebuds in (b). Also, our algorithm detects three people while ignoring more steep changes around the sun in (d).

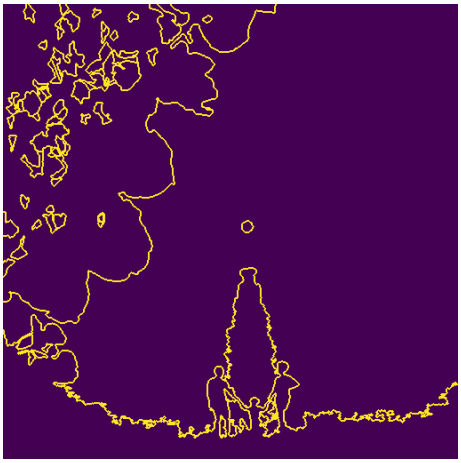
One more advantage of our segmentation algorithm is the robustness. Since the CSIQ dataset provides images with noise of various types and strengths, we can check the effect of the distortions on the segmentation results. Figure 3.14 shows the segmentation results on these damaged images. (a) is an image of blurred boundaries, (c) is the contrast changed image, and (e) is the image contaminated by additive Gaussian pink noises. The distortion level of all images is 3 which means that all distortions are quite visible to most people. Without any modification of parameters, we can obtain results of (b) and (f). And by slightly reducing bandwidth value, we can obtain a result of (d). It can be seen that the salient information of the image has been successfully preserved in the segmentation process, despite some damage to the information in images itself. While most of the algorithms have to go into time-consuming parameter readjustment when the environment of the image changes, our algorithms are relatively free from these problems.

(a) 1600.png



(b) Segmentation for (a)

(c) family.png



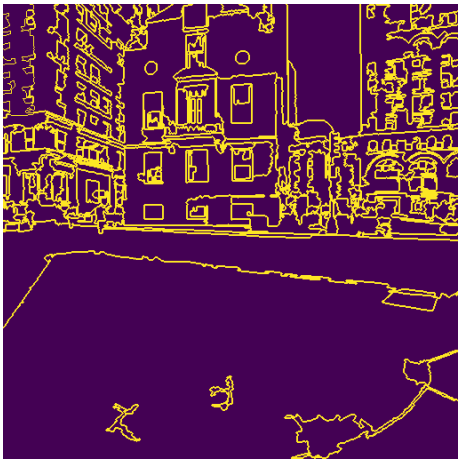
(d) Segmentation for (c)

(e) lady liberty.png

(f) Segmentation for (e)

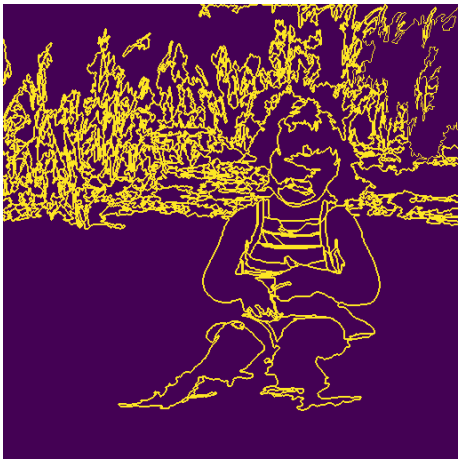
Figure 3.12: Segmentation Result of CSIQ Dataset(1)

(a) Boston.png



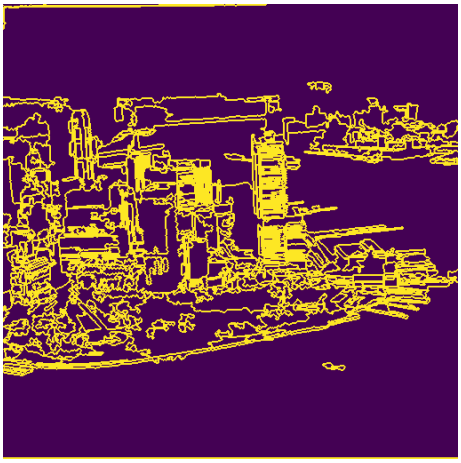
(b) Segmentation for (a)

(c) child swimming.png



(d) Segmentation for (c)

(e) aerial city.png



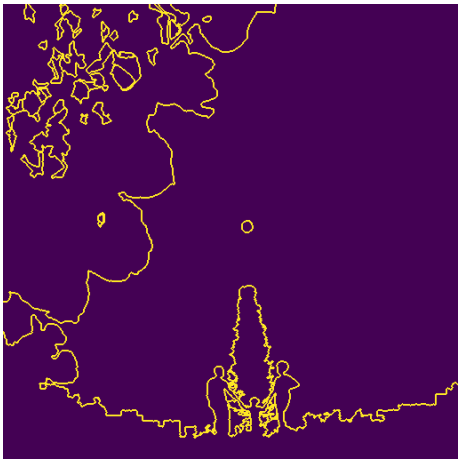
(f) Segmentation for (e)

Figure 3.13: Segmentation Result of CSIQ Dataset(2)

(a) Blurred Image

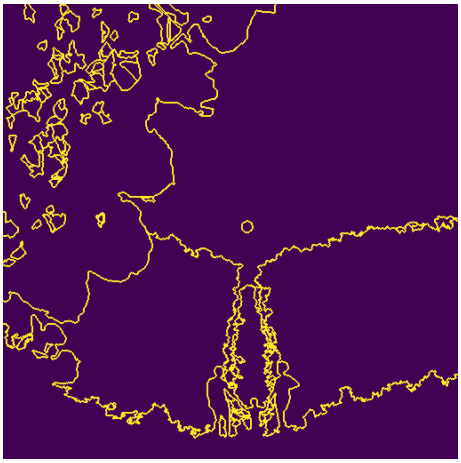
(b) Segmentation for (a)

(c) Contrast Changed Image



(d) Segmentation for (c)

(e) Contaminated by $\frac{1}{f}$ noise



(f) Segmentation for (e)

Figure 3.14: Segmentation Result of the Damaged 'family.png'

3.5.2 Experiment of Berkeley Dataset

Berkeley dataset [24] is generated to provide an empirical basis for research on image segmentation and boundary detection. This dataset is the collection of images with the hand-made ground truth for segmentation. The number of images is about 600 and each image has multiple manual segmentation results. Since it provides 'right answers' for segmentation, many deep learning networks for boundary detection [34] have been trained based on this dataset. Compared to the CSIQ dataset, this dataset is more suitable for segmentation. The images of the dataset usually have remarkable target objects in the center of them, so we can determine whether the segmentation results are good or poor.

Figure 3.15, 3.16, 3.17, and 3.18 show the results of our segmentation algorithm on Berkeley dataset. Since the Berkeley dataset has more images than the CSIQ dataset, we attach more results to confirm the effect of our method. Each result consists of three columns. The first column represents the original images and the second column is the manual ground truth of each image attached in the dataset. Since there are several ground truths, so we pick one of them. The third column is the results of our algorithm, all the results using the same bandwidth parameters to show the robustness of our algorithm. As shown in the figures, our algorithms are producing a result of quality. We are able to catch detailed boundaries much more accurately than before and it only takes an average of 20 seconds to produce these data, despite it may be changed depending on the images. We will cover the time issue in more detail in the next section.

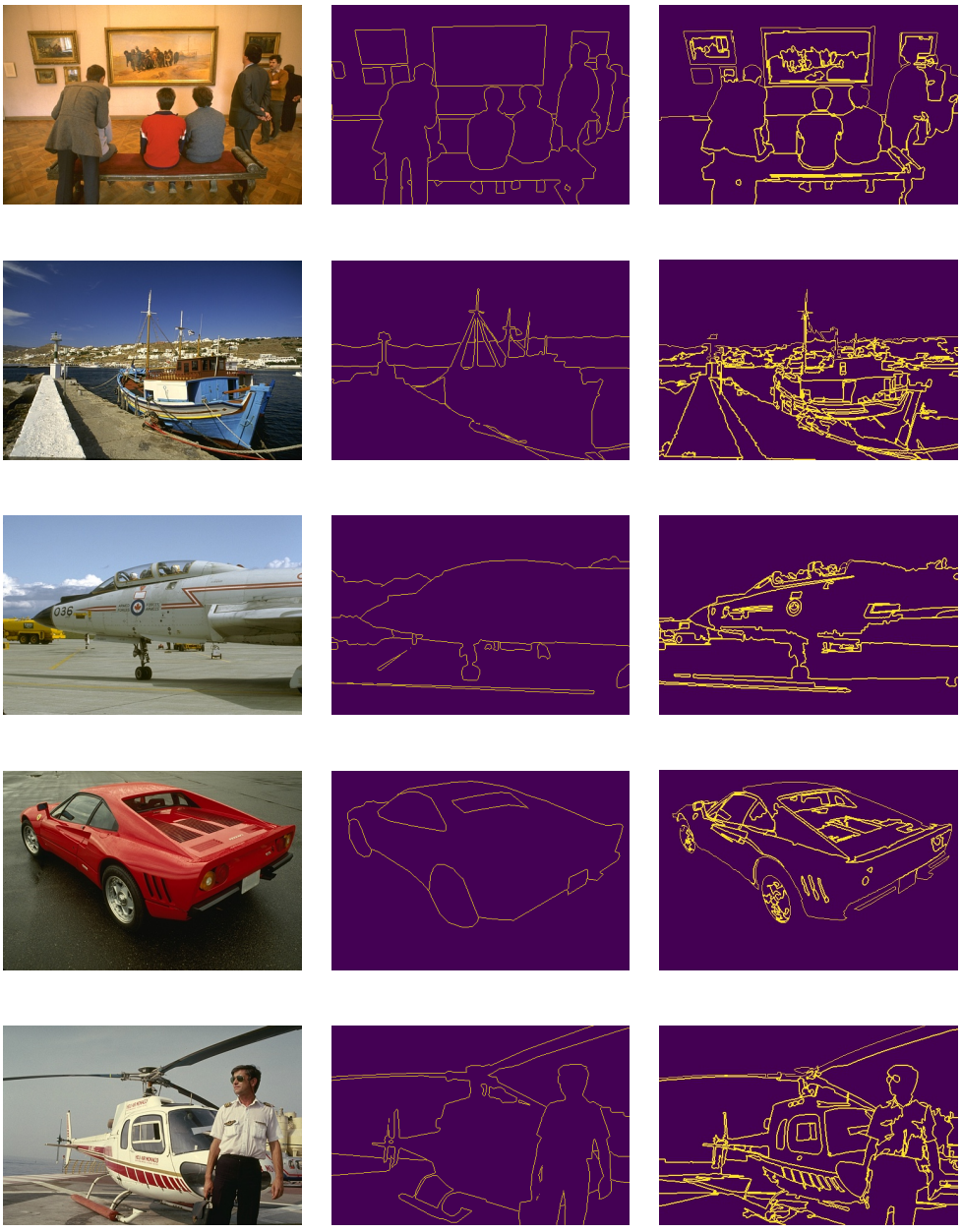


Figure 3.15: Segmentation Result of Berkeley Dataset(1)

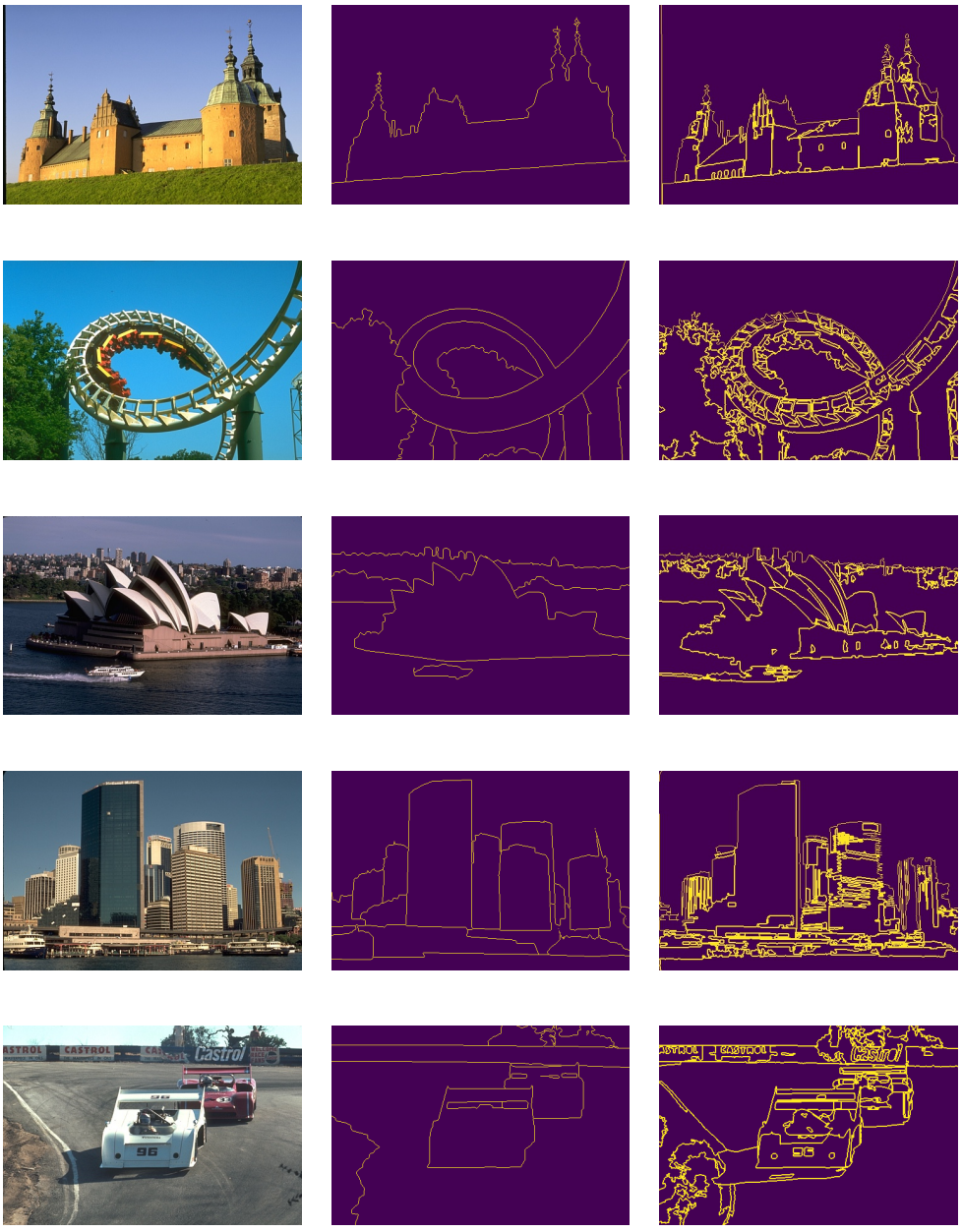


Figure 3.16: Segmentation Result of Berkeley Dataset(2)

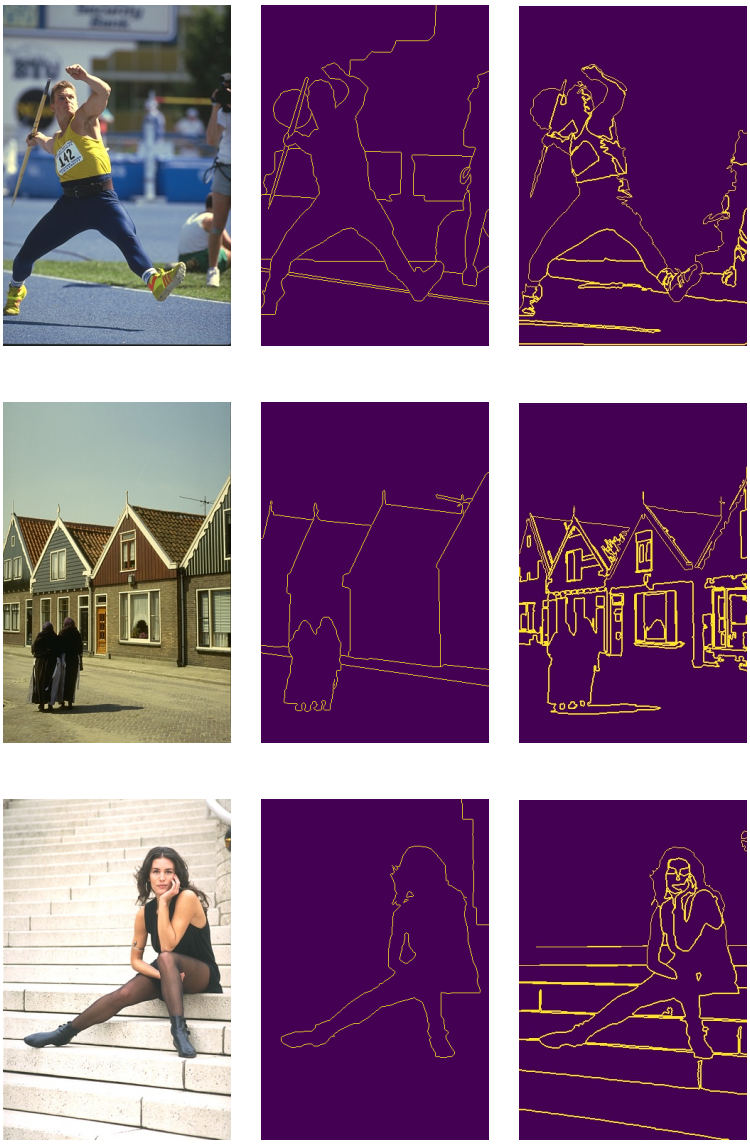


Figure 3.17: Segmentation Result of Berkeley Dataset(3)

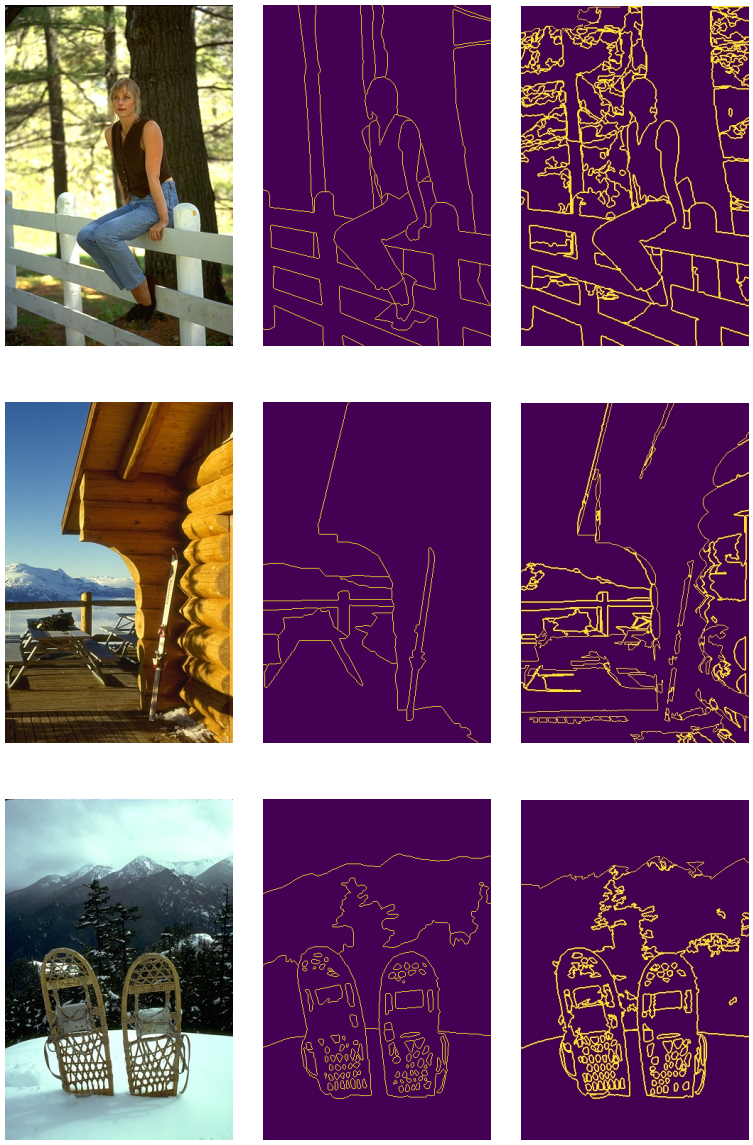
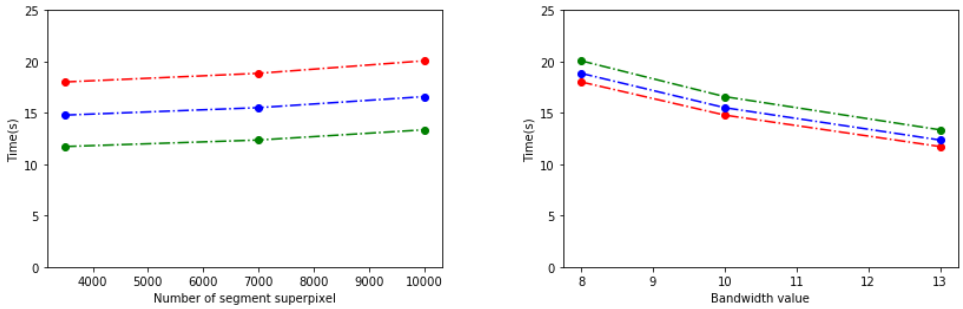


Figure 3.18: Segmentation Result of Berkeley Dataset(4)

3.5.3 Computational Time and Parameters



(a) Segment number and Time (b) Bandwidth value and Time

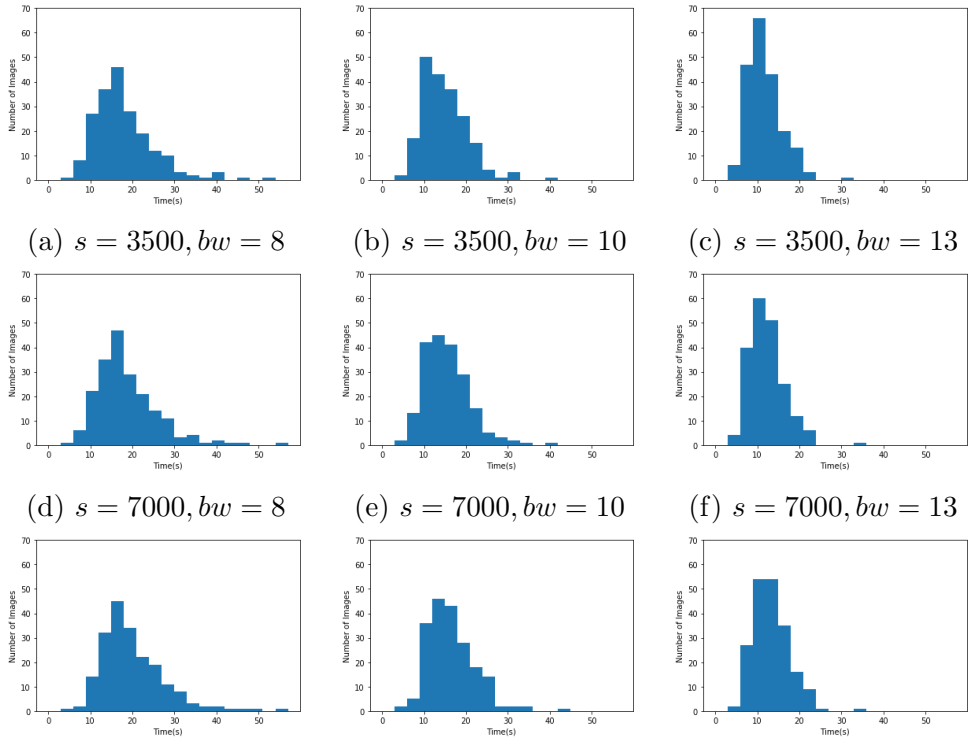
Figure 3.19: Time and the Parameters

Since Berkeley dataset has sufficiently many images, we can analyze the time issue of our algorithm. All the tests are performed on the test set of Berkeley dataset which is composed of 200 images.

Our algorithm has two main parameters, segment number, and bandwidth value. Two graphs in Figure 3.6 shows the relation between two parameters and the computation time. (a) illustrate the change of computation time as the number of segment superpixel increases for the different bandwidth values and (b) shows the change of computation time as the bandwidth value increases for the different segment superpixel number.

As (a) in the figure shows, increasing the number of the segment has nearly no effects on computation time. Therefore, there is no problem with the time to increase the segment superpixel number as needed. But if we use too many segment superpixel, our algorithm may be misled by the local noises or contamination, so desirable value for the number of segment superpixel is the minimum value that detects the smallest object in the images.

On the other hand, bandwidth value influences not only segmentation results but also the computation time, as shown in (b). If we use too low bandwidth value, then our S-Mean Shift result has extremely oversegmented and it takes a great time to process their boundaries. The histograms of time distribution are attached in Figure 3.20. As the bandwidth value shrinks, it can be seen that the number of outliers, which take extremely long, is



(a) $s = 3500, bw = 8$ (b) $s = 3500, bw = 10$ (c) $s = 3500, bw = 13$

(d) $s = 7000, bw = 8$ (e) $s = 7000, bw = 10$ (f) $s = 7000, bw = 13$

(g) $s = 10000, bw = 8$ (h) $s = 10000, bw = 10$ (i) $s = 10000, bw = 13$

Figure 3.20: Distribution of Segmentation Time for Berkeley Test Images

increasing. In this case, segmentation results are usually poor, so we should halt the algorithm and use higher bandwidth.

Chapter 4

Color Mapping Based on Superpixel

Color mapping or color transfer refer an algorithm that reconstruct target images using colors in the source images. Also, we may call a resulting function from above algorithm that maps the colors of target image to the colors of source image as color mapping. After transporting the color palette, the geometry of the produced result, such as boundaries of objects, must be consistent with the that of original target image as shown in the Figure 4.1. In addition, there should be no outlier in the generated image, in other word, looks *'Natural'* to the most of people. Computational time is an essential factor since these algorithms are usually slow. The quality of the color mapping algorithms is determined by taking all these factors into account. However, since there is no objective benchmark system, it heavily rely on the subjective score of the evaluator.

From the paper 'Color Transfer between Images' [27], many color mapping techniques have been developed. These methods utilize the color distribution of images and neighboring relations of the colors to change the color distribution while leaving no awkward visual outlier and give a similar impression as the source images. But these algorithms usually slow since it takes time to apprehend the structure of the images.

Figure 4.1: Example of the Color Mapping [30]

4.1 Color Mapping Problem and Superpixels

There have been attempts applying superpixels on the color mapping problem before. In [29], a relaxed optimal transport model is applied to color transfer using superpixel lower-level representation. Giraud et al. [30] proposed a novel color mapping method based on superpixels, called superpixel-based color transfer method(SCT) in 2017. Their method is based on the PatchMatch(PM) method [28], one of the approximate nearest neighbor(ANN) method.

The PatchMatch(PM) method computes correspondences between patches of two images A and B. It exploits the assumption that if patches are matched between A and B, then their respective adjacent neighbors should also match well. Such propagation, associated with a random selection of patch candidates, enables the algorithm to have a fast convergence towards good ANN.

We hereby introduce patchmatch method briefly. PM is composed of three steps. At first, one randomly assigns to each patch of A, a corresponding path in B. An iterative refinement process is then performed following a scan order(top left to bottom right) to refine the correspondences with the propagation and random search step. For a patch $A_i \in A$, the aim is to find the match $B_{(i)} \in B$ that minimizes a distance $D(A_i, B_{(i)})$, for instance the sum of squares differences of color intensities. During propagation, for each patch in A, the two recently processed adjacent patches are considered. Their matches in B are shifted to respect the relative positions in A, and the new candidates are tested for improvement. This propagation step is illustrated

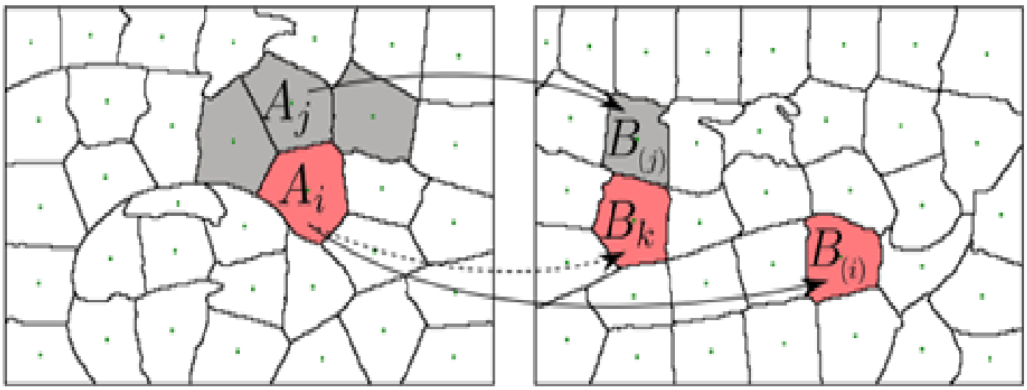


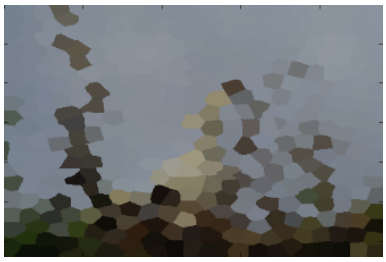
Figure 4.2: Propagation Step of the SCT [30]

in more detail in the Figure 4.2. The patch A_i(red) is currently matched to $B_{(i)}$. Its top-left adjacent neighbors A_j(gray) are considered to provide new candidate. A neighbor A_j is matched to $B_{(j)}$, which leads to the candidate B_k, the neighbor of $B_{(j)}$ with the most similar relative position to the one between A_i and A_j. Finally, the random search selects candidates around the current ANN in B to escape from local minima. This process is indispensable and we will show the effect later. If we perform sufficiently many iteration of second and third steps above, we will get satisfactory results.

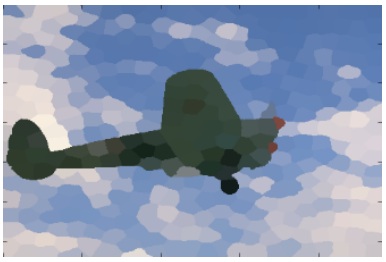
SCT attempt to combine the PM method and the idea of superpixel. This method adopts the superpixel segments as the patches of the PM method. The overall process of the SCT is shown in the Figure 4.3 in the next page. SLIC algorithm is applied to source image and target image and generates superpixels of them. In this process, (c) and (d) of the Figure 4.3 are obtained. And then, PM method above is applied using superpixels as patches. Random initial mapping of the PM method is shown in the (e) of the figure. (f) is the result of the propagation without random search step. Without random search, this algorithm can be stuck in the local minima. Therefore, propagation step of the PM should be with random search and the result is shown in the (g) of the figure. Finally, apply color fusion algorithm to (g) and final result (h) is obtained.

(a) Source Image

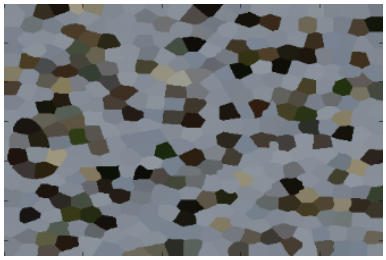
(b) Target Image



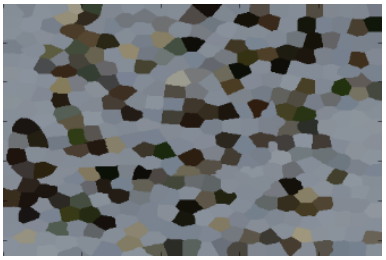
(c) Superpixelized Source Image



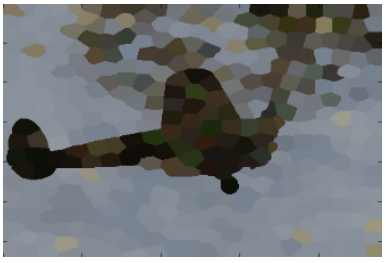
(d) Superpixelized Target Image



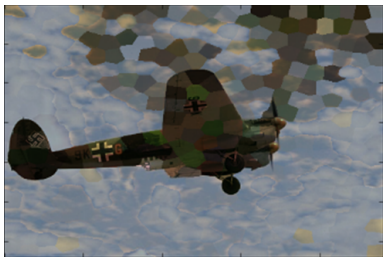
(e) Random Mapping



(f) Propagation without Random Search

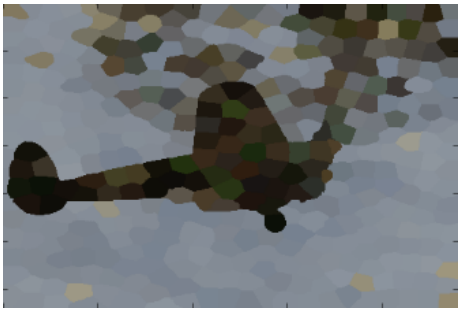


(g) Propagation with Random Search



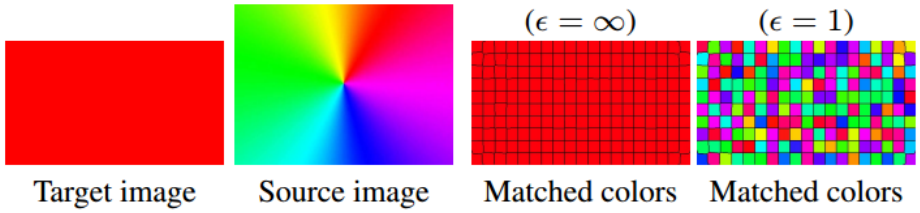
(h) Color Fusion

Figure 4.3: Process of the SCT with $\epsilon = 1$



(a) $\epsilon = 1$

(b) $\epsilon = 5$



(c) Effect of the ϵ

Figure 4.4: ϵ and the Result of the SCT

First result of the above process in shown in the (a) of the Figure 4.4. This is a very poor result, since objects that do not exist in the target image appear in the upper right corner. This phenomenon happens because the number of superpixels that make up objects is different in each image. For better result, we should set a parameter ϵ that defines the maximum number of selection of the same superpixel. The effect of the ϵ values is well illustrated in the (c) of the Figure 4.4 [30]. With the smaller ϵ, the color palette of the source image is better transported, and with the larger ϵ, the properties of the target image are better preserved. As shown in the (b) of the Figure 4.4, we get much better results by setting $\epsilon = 5$.

Finally, we should construct pixels of result image from the superpixel data. SCT adopts a non-local means fusion framework [31]. A superpixel A_i is described by the set of position X_i, and colors C_i of the contained pixels p, such that $A_i = [X_i, C_i] = [(\frac{x_i}{N_x}, \frac{y_i}{N_y}), (r_i, g_i, b_i)/255]$ with $N_x \times N_y$ the size of

Figure 4.5: Final Result of the the SCT [30]

image A. To compute the new color $A_t(p)$ of a pixel p, the weighted fusion of the matched colors is performed based on color and spatial similarity:

$$A_t(p) = \frac{\Sigma_j w(p, A_j)\bar{C}_{B(j)}}{\Sigma_j w(p, A_j)} \tag{4.1.1}$$

with $\bar{C}_{B(j)}$, the average color of the match of A_j in B, and $w(p, A_j)$ the weights of each pixels are defined based on the distance between the considered pixel $p \in A_j$ and the superpixel $A_j \in A$. This weight is computed similarly to a Mahalanobis distance:

$$w(p, A_j) = exp(-(p - \bar{A}_j)^T Q_i^- 1(p - \bar{A}_j) + \sigma(p)) \tag{4.1.2}$$

where $\sigma(p)$ sets the exponential dynamic and is set such that

$$\sigma(p) = \min_j (-(p - \bar{A}_j)^T Q_i^- 1(p - \bar{A}_j) + \sigma(p)) \tag{4.1.3}$$

and Q_i includes spatial and colorimetric covariances of the pixels in A_i:

$$Q_i = Q(A_i) = \begin{pmatrix} \sigma_s^2 Cov(X_i) & 0 \\ 0 & \sigma_c^2 Cov(C_i) \end{pmatrix} \tag{4.1.4}$$

Using this method, SCT generates final result as in the Figure 4.5.

4.2 Applying S-Mean Shift to Color Mapping

The SCT algorithm gives us two invaluable insights into the superpixel-based color mapping problem. The first is that if each piece is mapped 'correctly', there is no need to consider the boundary relation between the superpixels. The second is that without multiple selections of a single piece we cannot get good color mapping results.

But we also would like to point out two limitations of the SCT method. First, this algorithm heavily depends on randomness. There are two steps using random permutation in the algorithm, so it is impossible to reproduce the result perfectly. What's more, number of iteration needed to get proper result varies every time since it relies on a random search. It may not converge forever in the worst case. Since it is hard to find the appropriate terminate condition of iteration in the color mapping problem, we need a lot of iteration to guarantee the decent results. This makes the computational time much longer. Second, the result of the algorithm heavily depends on ϵ value as mentioned above. But this algorithm does not provide any clues about the appropriate value of ϵ. We should perform the SCT algorithm several times with varying ϵ and compare the results subjectively. This is not suited for large datasets.

We try to overcome these limitations by applying S-Mean Shift to the SCT algorithm. The RCs of each image which can be obtained by S-Mean Shift algorithm are utilized in our method. If S-Mean Shift is applied to both source and target images, we will get color clusters of each image. Let N_s and N_t be the cluster number of the source image and target image, respectively. Also, we denote the center of ith cluster in the source image and the number of superpixels in that cluster C_s^i and N_s^i. C_t^i and N_t^i are defined in an analogous way for the target image. Because it is natural to map superpixels in the source image of the same cluster to the same cluster in the target image, we can simplify the problem as finding the mapping

function $f : \{1, 2, \cdots N_t\} \rightarrow \{1, 2, \cdots N_s\}$ such that:

$$\text{Minimize } \Sigma_{k=1\cdots N_t} ||C_t^k - C_s^{f(k)}||_2 \cdot N_t \text{ satisfying } \Sigma_{f(k)=i} N_t^k < \epsilon \cdot N_s^i \quad (4.2.1)$$

if we have given ϵ, or

$$\text{Minimize } \Sigma_{k=1\cdots N_t} ||C_t^k - C_s^{f(k)}||_2 \cdot N_t + \lambda\epsilon \text{ where } \epsilon = \max_i \frac{\Sigma_{f(k)=i} N_t^k}{N_s^i} \quad (4.2.2)$$

Now we have an easy-to-calculate energy function for the color mapping problem and some guidelines for the ϵ value. If $\epsilon < \frac{\max_i N_t^i}{\max_i N_c^i}$, there are no possible solutions. Set $\epsilon = \infty$ and minimize $\Sigma_{k=1\cdots N_t} ||C_t^k - C_s^{f(k)}||_2 \cdot N_t$. If $\epsilon > \epsilon_\infty = \max_i \frac{\Sigma_{f(k)=i} N_t^k}{N_s^i}$, we get same result as $\epsilon = \infty$. Since second equation is hard to solve and there are few ϵ available, we will use the first one in this paper.

For the case in Figure 4.3, and if we set $\epsilon = \infty$, our energy function has a value of 360.25 and $\frac{\Sigma_{f(k)=i} N_t^k}{N_s^i} = 2.58$. Therefore our energy function is 360.25 for $\epsilon > 3$. If we set $\epsilon = 2$, our energy function increases to 394.55 and when $\epsilon = 1$ there is no possible solution. So we may choose $\epsilon = 2$ or $\epsilon = 3$ and both choices have their grounds.

Let S_s^i is the set of the superpixels in the source image whose cluster label is i. We can also define S_t^i in the analogous way for the superpixels in the target image. After we find $f(k)$ for each $k \in \{1, 2, \cdots, N_t\}$, we should find the map g_k from S_t^k to $S_s^{f(k)}$. If we map all the elements to S_t^k to the $C_s^{f(k)}$, this algorithm is extremely sensitive to bandwidth parameter of S-Mean Shift algorithm.

We try to find f and g_k for each k based on greedy algorithm. Algorithm below is described using the notations for g_k. To avoid the loop, mapping never moves to the more similar superpixels.

Algorithm 5 Finding Best Mapping

Map each cluster center S_t^k to the most similar superpixel in $S_s^{f(k)}$.
Denote this mapping as g_k
while True **do**
    Generate the set S_ϵ of s such that $|\{x \in S_t^k | g_k(x) = s\}| > \epsilon$.
    if $|S_\epsilon| == 0$ **then**
        break
    end if
    For each s in S_ϵ, define $X_s = \{x \in S_t^k | g_k(x) = s\}$.
    Calculate the increment of the distance for all x in X_s for each s
    if we change the value of $g_k(x)$ to the next similar superpixels in $S_s^{f(k)}$.
    Except ϵ elements with highest increment for each X_s
    move all the superpixels S_t^k in to the next similar superpixels.
end while

(a) Color mapping using superpixel

(b) Color Fusion Applied

Figure 4.6: Result of the Algorithm(1)

(a) in Figure 4.6 is the result of our algorithm. This method is much faster since there are no random searches and all the iterative processes are excluded. Each mapping takes about 10s using Python code. (b) is the result after the color fusion is applied same as the SCT. There is one more example of our algorithm in the next page.

(a) Source Image

(b) Target Image

(c) Color Mapping

(d) Color Fusion

Figure 4.7: Result of the Algorithm(2)

Chapter 5

Conclusion

In this paper, we devise a robust image analyzing method based on pivot colors of the images. The superpixelwise mean shift algorithm which perform the mean shift algorithm on the representative values of the superpixels, overcome the speed limitation of the mean shift algorithm and can handle extremely complicated natural images.

In chapter 3, we proposed an adaptive image segmentation method based on the S-mean shift algorithm. This method produces image segmentation results quite analogous to the human visual system and can also handle extremely complex images that are difficult to respond by conventional methods. Since it shows higher quality than mass-produced ground truth while it does take tolerable time for generation and needs no pre-training, this approach is also suitable for creating training data for various deep learning networks. However, this method tends to over-segment images than desirable results. This may occur since we used only color information not semantic information of objects in the images. But it may be caused by insufficient utilization of color information, so further study is needed about that.

There is also regret about the time issues, which is that most of the computation time is spent collecting small and meaningless pieces rather than the core algorithm. If we find a smarter way to achieve this work, computation time will be greatly reduced.

Next in chapter 4, we try to improve the disadvantage of the SCT method by combining our S-Mean shift algorithm. Our approach has reduced the un-

certainty of the existing method and has accelerated the speed of the conventional method. We also pointed out the difficulty in choosing parameters of the existing method and proposed the ground for that.

With its speed, stability, and similarity to the human cognitive system, the S-Mean shift has many possibilities beyond what is stated here. If further research is continued, I believe that more applications will be possible based on this approach.

Bibliography

[1] Z. Wu and R. Leahy, *An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15, no.11(1993), 1,101-1,113.

[2] J. Shi and J. Malik , *Normalized Cuts and Image Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.22, No.8 (2000), 888-905.

[3] D. Comaniciu and P. Meer, *Mean Shift: A Robust Approach toward Feature Space Analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.24, No.5(2002), 603-619

[4] P. F. Felzenszwalb and D. P. Huttenlocher, *Efficient Graph-Based Image Segmentation*, International Journal of Computer Vision Vol.59, No.2(2004), 167–181

[5] D. Comaniciu and P. Meer, *Mean Shift: A Robust Approach toward Feature Space Analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.24, No.5(2002), 603–619

[6] Y. Cheng, *Mean Shift, mode seeking, and clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.17(1995), 790–799

[7] J. Chen, Z. Li, and B. Huang*Linear Spectral Clustering Superpixel*, IEEE Transactions on Image Processing Vol.26, No.7(2017), 3317-3330

BIBLIOGRAPHY

[8] R. Achanta, A. Shaji, K. Smith, A. Lucch, P. Fua, and S. Su¨sstrunk, *SLIC: Superpixels Compared to State-of-the-Art Superpixel methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.34, No.11(2012), 2274-2281.

[9] J. Yao, M. Boben, S. Fidler and R. Urtasun, *Real-Time Coarse-to-fine Topologically Preserving Segmentation*, IEEE Conference on Computer Vision and Pattern Recognition(2015), 2947-2955.

[10] D. Stutz, A. Hermans, and B. Leibe, *Superpixels: An Evaluation of the state-of-the-art*, Computer Vision and Image Understanding 166(2018), 1–27.

[11] D. Martin, C. Fowlkes, and J. Malik, *Learning to detect natural image boundaries using local brightness, color and texture cues*, IEEE Transactions on Pattern Analysis and Machine Intelligence Vol.26 No.5(2004), 530–549.

[12] P. Neubert, and P. Protzel, *Superpixel benchmark and comparison*, Forum Bildverarbeitung(2012).

[13] A. P. Moore, S. J. D. Prince, J. Warrell, U.Mohammed and G. Jones, *Superpixel lattices*, IEEE Conference on Computer Vision and Pattern Recognition(2008), 1–8.

[14] Z. Wang, and A.C. Bovik, *A universal image quality index*, IEEE Signal Processing Letters Vol.9 (2002), 81–84.

[15] Z. Wang, A.C. Bovik, H. R. Sheikh, and E. P. Simoncelli *Image Quality Assessment: From Error Visibility to Structural Similarity*, IEEE Transactions on Image Processing Vol.13 No.4(2004), 600–612.

[16] Z. Wang, E. P. Simoncelli and A.C. Bovik, *Multi-scale Structural Similarity for Image Quality Assessment*, Conference Record of the Asilomar Conference on Signals, Systems and Computers Vol.2 No.2(2003), 1398 - 1402

[17] A. Liu, and W. Lin, *Image Quality Assessment Based on Gardient Similarity*, IEEE Transactions on Image Processing Vol.21 No.4(2012), 1500–1512.

[18] W. Sun, Q. Liao, J. Xue and F. Whou, *SPSIM: A Superpixel-Based Similarity Index for Full-Reference Image Quality Assessment*, IEEE Transactions on Image Processing Vol.27 No.9(2018), 4232–4244.

[19] M. Wang, and T. Blu, *Generalized YUV interpolation of CFA images*, IEEE International Conference On Image Processing(2010), 1909–1912.

[20] H. R. Sheikh, Z. Wang, L.Cormack and A. C. Bovik, *LIVE Image Quality Assessment Database Release*, http://live.ece.utexas.edu/research/quality.

[21] E. C. Larson, and D. M. Chandler, *Most apparent distortion: full-reference image quality assessment and the role of strategy*, Journal of Electronic Imaging(2010), 19(1).

[22] N. Ponomarenko, V. Lukin, A. Zelensky, K. Egiazarian, M. Carli, and F. Battisti, *A Database for Evaluation of Full-Reference Visual Quality Assessment Metrics*, Advances of Modern Radioelectronics Vol.10(2009), 30-45

[23] N. Ponomarenko, O. Ieremeiev, V. Lukin, K. Egiazarian, L. Jin, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C.-C. J. Kuo, *A New Color Image Database TID2013: Innovations and Results*, Proceedings of ACIVS, Poznan, Poland, Oct.(2013), 402-413

[24] D. Martin, C. Fowlkes, D. Tal and J. Malik, *A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics*, 8th International Conference of Computer Vision(2001), 416-423

[25] R. Tibshirani, G. Walther, T.Hastie, *Estimating the Number of Clusters in a Data Set Via the Gap Statistic*, Journal of the Royal Statistical Society Series(2001), B. 63. 411-423

[26] A. Tremeau, and P. COlantoni, *Regions adjacency graph applied to color image segmentation* , IEEE Transactions on Image Processing Vol.9 No.4(2000), 735-744

[27] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, *Color Transfer between Images*, IEEE Computer Graphics and Applications, Vol.21, No.5(2001), 34–41

[28] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, *PatchMatch: A randomized correspondence algorithm for structural image editing*, ACM Transactions on Graphics, Vol.28, No.3(2009)

[29] J. Rabin and N. Papadakis, *Non-convex relaxation of optimal transport for color transfer*, NIPS 2014 Workshop on Optimal Transport and Machine Learning(2014)

[30] R. Giraud, V. Ta, and N. Papadakis, *Superpixel-Based Color Transfer*, IEEE International Conference on Image Processing(2017), 34–41

[31] A. Buades, B. Coll, and J.-M. Morel, *A Non-local Algorithm for Image Denoising*, IEEE Computer Vision and Pattern Recognition, Vol.2(2005), 60–65

[32] H. Ko, and J. Ding, *Adaptive Growing and Merging Algorithm for Image Segmentation*, 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)(2016)

[33] A. J. Dunnings and T. P. Breckon, *Experimentally defined Convolutional Neural Network Architecture Variants for Non-temporal Real-time Fire Detection*, 2018 25th IEEE International Conference on Image Processing(2018)

[34] I. Kokkinos, *Pushing the boundaries of boundary detection using deep learning*, 2016 International Conference on Learning Representations(2016)

국문초록

인간과 비슷하게 동작하게 인공 지능의 개발은 이미지 프로세싱 분야에서도 중요한 이슈 중에 하나이다. 이러한 추세에서 픽셀을 기반으로 이미지를 다루는 고전적인 방법들은 여러 가지 한계를 보여주는데, 가장 큰 이유는 인간은 개별 픽셀이 가지는 정보에 큰 관심을 주지 않기 때문이다. 많은 연구가 보여주듯이 인간을 이미지를 의미를 가지는 수많은 '덩어리' 들의 복합적인 결합으로 보는 경향이 있으며 다양하고 복잡한 이미지를 다루기 위해서는 픽셀보다는 이러한 '덩어리'를 기반으로 이미지를 파악할 필요가 있다.

이 논문에서는 이미지에서 이러한 '덩어리' 역할을 할 수 있는 슈퍼픽셀을 다룬다. 앞부분에서는 먼저 다양한 슈퍼픽셀 생성 기법들과 그들의 장단점을 소개하고 이미지 평가 분야에서의 결과를 바탕으로 슈퍼픽셀이 이미지를 다루는데 얼마나 효과적인지를 보이겠다. 그 다음에 우리는 이미지의 기조 색상을 바탕으로 한 새로운 이미지 분석 방법을 제시하고 그 기조 색상을 구하기 위해서 Superpixelwise Mean Shift라는 새로운 방법론을 제시하였다. 이 방식은 Mean shift procedure와 슈퍼픽셀의 대표값을 결합시킨 방식으로 매우 빠르고 확고하다. 뒤의 장에서는 이 방법론을 이미지 분할 문제와 색 이동 문제에서 적용하고 그 결과를 보이겠다.

주요어휘**:** 이미지 처리, 슈퍼픽셀, 클러스터링, 이미지 분할, 색 이동
학번**:** 2015-30088

감사의 글

제가 학업을 시작하고부터, 여기 서울대학교 수리과학과에서 박사 학위를 받기까지 너무나도 많은 분에게 이루 말할 수 없을만큼 도움을 받았다. 초등학교 시절 수학에 재능이 있다며 진로를 추천해 주신 담임 선생님, 중학교 시절 수학경기반 설립을 도와주신 교장 선생님, 어려웠던 가정 형편을 고려하여 학원비를 받지 않으셨던 원장 선생님,여기에 저를 도와주신 분들을 모두 적는다면 이 페이지의 여백이 아무리 많아도 모자랄 것이다. 그렇기에 여기에 적는 것은 도움을 주신 그분들 중 극히 일부이며 비록 여기에는 적지 못하더라도 많은 분들이 주신 은혜를 결코 잊지 않을 것이다.

이 자리를 빌어서 가장 먼저 그리고, 가장 깊이 감사드리고 싶은 분은 어머니이다. 어려서부터 가정 형편이 그리 좋지 못하였고, 특히 제가 고등학교 때 부모님이 헤어지시면서 가족 모두가 힘든 시기를 맞았었는데 언제나 재정적, 정신적으로 버팀목이 되어주신 어머니가 아니었으면 절대 이 자리까지 올 수 없었을 것이다. 또, 어머니와 저,제 동생을 친자식처럼 아껴주시고 힘들게 일하시면서도 제가 잘 되기만을 바라신 아버지께도 깊은 감사를 드린다.

그 다음으로 지도 교수님인 강명주 교수님께 감사 인사를 올리고 싶다. 박사 입학 과정에서 착오가 생겨서 지도 교수를 찾지 못해 서울대학교에 지원조차 하지 못할 상황에서 구원의 손길을 내밀어 주셨고, 지난 5년 간 언제나 더 좋은 성과를 낼 수 있도록 끊임없이 이끌어주셨다. 이 연구실에서 프로젝트를 수행하면서 쌓은 경험과 인연이 졸업 후의 진로를 결정하는데도 큰 도움이 되었으니, 강명주 교수님이야말로 제 인생의 방향을 잡아주신 최고의 은사님으로 평생 기억할 것이다.

마지막으로 같은 분야의 선배로 연구에 조언을 아끼지 않은 강명민 교수님과 나한울 박사님, 연구실의 큰 형님 노릇을 해주신 이병준 교수님, 그 밖에 같은 연구실의 모두에게 감사드린다. 올해를 마지막으로 연구실을 떠나게 되지만 지난 5년 간은 제 인생의 보물같은 시간으로 영원히 잊지 못할 것이다. 회사에 들어가서도 최대한 열심히 하여 만약 후배가 저와 같은 길을 걸으려 할 때, 발목을 잡는 선배가 되지 않도록 노력할 것이다.